IEC 62056-6-2

Edition 1.0 2013-05

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Electricity metering data exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes

Échange de données dans les équipements de comptage de l'énergie électrique – La suite DLMS/COSEM – Partie 6-2: Classes d'interfaces COSEM

IEC 62056-6-2:2013

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you to find IEC publications by a variety of criteria (reference number, text, technical committee,…).
It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available on-line and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Liens utiles:

Recherche de publications CEI - www.iec.ch/searchpub

La recherche avancée vous permet de trouver des publications CEI en utilisant différents critères (numéro de référence, texte, comité d'études,…).
Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 62056-6-2

Edition 1.0 2013-05

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Electricity metering data exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes

Échange de données dans les équipements de comptage de l'énergie électrique – La suite DLMS/COSEM – Partie 6-2: Classes d'interfaces COSEM

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XH

ICS 17.220; 35.110; 91.140.50

ISBN 978-2-83220-822-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE –

Part 6-2: COSEM interface classes

FOREWORD

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

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

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

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

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

[1] Device Language Message Specification.

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

This edition cancels and replaces IEC 62056-62 published in 2006. It constitutes a technical revision.

The significant technical changes with respect to IEC 62056-62 are listed in Annex A.

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

FDIS	Report on voting
13/1525/FDIS	13/1543/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.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all the parts in the IEC 62056 series, published under the general title *Electricity metering data exchange – The DLMS/COSEM suite*, can be found on the IEC website.

Future standards in this series will carry the new general title as cited above. Titles of existing standards in this series will be updated at the time of the next edition.

The numbering scheme has changed from IEC 62056-XY to IEC 62056-X-Y. For example IEC 62056-62 becomes IEC 62056-6-2.

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

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

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

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

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

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

To allow further analysis of information, for the purposes of billing, load-, customer- and contract management, it is necessary to uniquely identify data items, whether collected manually or automatically, via local or remote data exchange, in a manufacturer-independent way. The definition of identification codes to achieve this – the OBIS codes – is based on DIN 43863-3:1997, *Electricity meters – Part 3: Tariff metering device as additional equipment for electricity meters – EDIS – Energy Data Identification System.*

The COSEM model represents the meter as a server – see 4.7 – used by client applications that retrieve data from, provide control information to, and instigate known actions within the meter via controlled access to the attributes and specific methods of objects making up the server interface. This client may be supporting the business processes of utilities, customers, meter operators, or meter manufacturers.

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

ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE –

Part 6-2: COSEM interface classes

1 Scope

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

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

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

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

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

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

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

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

Draft IEC 62056-3-1:—, *Electricity metering data exchange – The DLMS/COSEM suite – Part 3-1: Use of local area networks on twisted pair with carrier signalling*[2]

IEC 62056-46:2002, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol*
Amendment 1:2006

---

2 To be published.

IEC 62056-5-3:—, *Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: DLMS/COSEM application layer*[3]

IEC 62056-6-1:—, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-1: Object identification system (OBIS)*[4]

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

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

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

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

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

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

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

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

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

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

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

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

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

---

3 To be published simultaneously with this part of IEC 62056.

4 To be published simultaneously with this part of IEC 62056.

STD 0005:1981, Internet Engineering Task Force (IETF). *Internet Protocol. Also: RFC0791 (RFC0792, RFC0919, RFC0922, RFC0950, RFC1112). Intellectual Property Rights in IETF Technology*. Edited by Jon Postel. September 1981. Available from: http://www.faqs.org/rfcs/std/std5.html

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

NOTE See also the Bibliography.

3 Abbreviations

AA	Application Association
AARE	A-Associate Response – an APDU of the ACSE
AARQ	A-Associate Request– an APDU of the ACSE
ACSE	Association Control Service Element
AL	Application Layer
AP	Application Process
APDU	Application Layer Protocol Data Unit
ASE	Application Service Element
A-XDR	Adapted Extended Data Representation
base_name	The short_name corresponding to the first attribute (“logical_name”) of a COSEM object
CHAP	Challenge Handshake Authentication Protocol
class_id	COSEM interface class identification code
COSEM	Companion Specification for Energy Metering
COSEM object	An instance of a COSEM interface class
CtoS	Client to Server challenge
DHCP	Dynamic Host Configuration Protocol
DLMS	Device Language Message Specification
DLMS UA	DLMS User Association
DNS	Domain Name Server
EAP	Extensible Authentication Protocol
GCM	Galois/Counter Mode, an algorithm for authenticated encryption with associated data
GMT	Greenwich Mean Time. Replaced by Coordinated Universal Time (UTC).
GPS	Global Positioning System
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HART	Highway Addressable Remote Transducer see http://www.hartcomm.org/ (in relation with the Sensor manager interface class)
HDLC	High-level Data Link Control
HLS	High Level Security
IANA	Internet Assigned Numbers Authority
IC	Interface Class

IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
IT	Information Technology
ISO	International Organization for Standardization
KEK	Key Encryption Key
LCP	Link Control Protocol
LLC	Logical Link Control (Sublayer)
LLS	Low Level Security
LN	Logical Name (used in relation to referencing attributes and methods of COSEM objects)
LSB	Least Significant Bit
m	mandatory
MD5	Message Digest Algorithm 5
MID	Measuring Instruments Directive 2004/22/EC
MSB	Most Significant Bit
o	optional
OBIS	OBject Identification System
PAP	Password Authentication Protocol
PDU	Protocol Data Unit
PLMN	Public Land Mobile Network
PIN	Personal Identity Number
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
RDR	Reply Data on Request (used in IEC 61334-3-32)
REJ PDU	Reject Protocol Data Unit
ROHC	Robust Header Compression
SAP	Service Access Point
SHA-1	Secure Hash Algorithm
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SN	Short Name (used in relation to referencing attributes and methods of COSEM objects)
StoC	Server to Client Challenge
UI PDU	Unnumbered Information Protocol Data Unit
UTC	Coordinated Universal Time

4 Basic principles

4.1 General

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

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

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

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

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

Figure 1 illustrates these terms by means of an example:

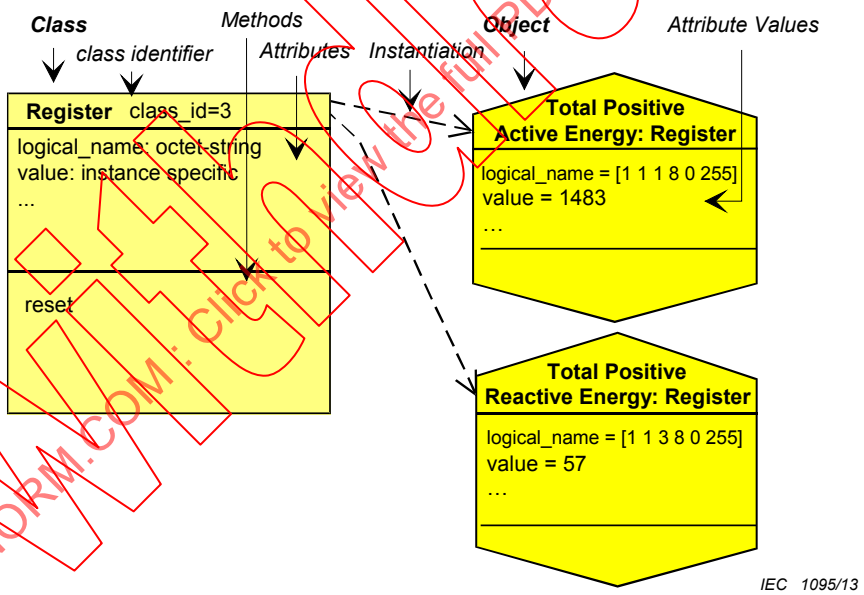


Figure 1 – An interface class and its instances

The IC “Register” is formed by combining the features necessary to model the behaviour of a generic register (containing measured or static information) as seen from the client (data collection system, hand held terminal). The contents of the register are identified by the attribute “logical_name”. The logical_name contains an OBIS identifier (see IEC 62056-6-1). The actual (dynamic) content of the register is carried by its “value” attribute.

Defining a specific meter means defining several specific objects. In the example of Figure 1, the meter contains two registers; i.e. two specific instances of the IC “Register” are instantiated. Through the instantiation, one COSEM object becomes a “total, positive, active energy register” whereas the other becomes a “total, positive, reactive energy register”.

NOTE The COSEM interface objects (instances of COSEM ICs) represent the behaviour of the meter as seen from the “outside”. Therefore, modifying the value of an attribute – for example resetting the value of a register – is

always initiated from the outside. Internally initiated changes of the attributes – for example updating the value of a register – are not described in this model.

4.2 Referencing methods

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

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

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

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

Where:

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

Using short names (SN referencing): This kind of referencing is intended for use in simple devices. In this case, each attribute and method of a COSEM object is identified with a 13-bit integer. The syntax for the short name is the same as the syntax of the name of a DLMS named variable. IEC 61334-4-41:1996, 4.4.6 and IEC 62056-5-3:—, Clause 8 apply.

4.3 Reserved base_names for special COSEM objects

In order to facilitate access to devices using SN referencing, some short_names are reserved as base_names for special COSEM objects. The range for reserved base_names is from 0x**FA00** to 0x**FFF8**. The following specific base_names are defined, see Table 1:

Table 1 – Reserved base_names for SN referencing

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

4.4 Class description notation

This Subclause 4.4 describes the notation used to define the ICs.

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

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

Class name Describes the interface class (for example “Register”, “Clock”, “Profile generic”…)

Cardinality Specifies the number of instances of the IC within a logical device (see 4.8).

value The IC shall be instantiated exactly “value” times.

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

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

Class_id-s from 0 to 8 191 are reserved to be specified by the DLMS UA.

Class_id-s from 8 192 to 32 767 are reserved for manufacturer specific ICs.

Class_id-s from 32 768 to 65 535 are reserved for user group specific ICs.

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

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

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

Attribute(s) Specifies the attribute(s) that belong to the IC.

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

	(static)	Classifies an attribute, which is not updated by the meter itself (for example configuration data). NOTE 1 There are some attributes, which may be either static or dynamic depending on the application. In these cases this property is not indicated.
logical_name	octet-string	The logical name is always the first attribute of an IC. It identifies the instantiation (COSEM object) of this IC. The value of the logical_name conforms to OBIS; see Clause 6 and IEC 62056-6-1.
Data type	Defines the data type of an attribute; see 4.5.	
Min.	Specifies if the attribute has a minimum value. *X* The attribute has a minimum value. <*empty*> The attribute has no minimum value.	
Max.	Defines if the attribute has a maximum value. *X* The attribute has a maximum value. <*empty*> The attribute has no maximum value.	
Def.	Specifies if the attribute has a default value. This is the value of the attribute after reset. X The attribute has a default value. <*empty*> The default value is not defined by the IC definition.	
Short name	When Short Name (SN) referencing is used, each attribute and method of object instances has to be mapped to short names. The base_name x of each object instance is the DLMS named variable the logical name attribute is mapped to. It is selected in the implementation phase. The IC definition specifies the offsets for the other attributes and for the methods.	
Specific method(s)	Provides a list of the specific methods that belong to the object. Method Name () The method has to be described in the subsection "Method description".	
m/o	Defines if the method is mandatory or optional. *m (mandatory)* The method is mandatory. *o (optional)* The method is optional.	

Attribute description

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

Method description

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

NOTE 2 Services for accessing attributes or methods by the protocol are described in IEC 62056-5-3:—, 6.6 to 6.15.

Selective access

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

4.5 Common data types

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

Table 2 – Common data types

Type description	Tag [a]	Definition	Value range
-- simple data types			
null-data	[0]		
boolean	[3]	Boolean	TRUE or FALSE
bit-string	[4]	An ordered sequence of boolean values	
double-long	[5]	Integer32	-2 147 483 648... 2 147 483 647
double-long-unsigned	[6]	Unsigned32	0...4 294 967 295
octet-string	[9]	An ordered sequence of octets (8 bit bytes)	
visible-string	[10]	An ordered sequence of ASCII characters	
	[11]	Tag of the “time” type in IEC 61334-4-41, not usable in DLMS/COSEM. See tag [27]	
UTF8-string	[12]	An ordered sequence of characters encoded as UTF-8	
bcd	[13]	binary coded decimal	
integer	[15]	Integer8	-128...127
long	[16]	Integer16	-32 768...32 767
unsigned	[17]	Unsigned8	0...255
long-unsigned	[18]	Unsigned16	0...65 535
long64	[20]	Integer64	$-2^{63}...2^{63}-1$
long64-unsigned	[21]	Unsigned64	$0...2^{64}-1$
enum	[22]	The elements of the enumeration type are defined in the “Attribute description” section of a COSEM IC specification.	0...255
float32	[23]	OCTET STRING (SIZE(4))	For formatting, see 4.6.2.
float64	[24]	OCTET STRING (SIZE(8))	
date_time	[25]	OCTET STRING SIZE(12))	For formatting, see 4.6.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	

Type description	Tag [a]	Definition	Value range
-- complex data types			
array	[1]	The elements of the array are defined in the “Attribute description” section of a COSEM IC specification.	
structure	[2]	The elements of the structure are defined in the “Attribute description” section of a COSEM IC specification.	
compact array	[19]	The elements of the compact array are defined in the “Attribute description” section of a COSEM IC specification.	
-- CHOICE		For some attributes of some COSEM interface objects, the data type may be chosen at COSEM object instantiation, in the implementation phase of the COSEM server. The server shall always send back the data type and the value of each attribute, so that together with the logical name, an unambiguous interpretation is ensured. The list of possible data types is defined in the “Attribute description” section of a COSEM IC specification.	
a The tags are as defined in IEC 62056-5-3:—, Clause 8.			

4.6 Data formats

4.6.1 Date and time formats

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

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

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

date OCTET STRING (SIZE(5))
{
year highbyte,
year lowbyte,
month,
day of month,
day of week
}
year: interpreted as long-unsigned
range 0…big
0xFFFF = not specified
year highbyte and year lowbyte reference the 2 bytes of the long-unsigned

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

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

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

For dayOfMonth and dayOfWeek:

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

The elements dayOfMonth and dayOfWeek shall be interpreted together:

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

EXAMPLE 1 year = 0xFFFF, month = FF, dayOfMonth = 0xFE, dayofWeek = 0xFF: last day of the month in every year and month;

EXAMPLE 2 year = 0xFFFF, month = FF, dayOfMonth = 0xFE, dayofWeek = 0x07: last Sunday in every year and month;

EXAMPLE 3 year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayofWeek = 0x07: last Sunday in March in every year;

EXAMPLE 4 year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayofWeek = 0x07: first Sunday in March in every year;

EXAMPLE 5 year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayofWeek = 0x05: fourth Friday in March in every year;

EXAMPLE 6 year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayofWeek = 0x07: fourth Sunday in October in every year;

EXAMPLE 7 year = 0x07D9, month = 0x01, dayOfMonth = 0x13, dayofWeek = 0x01: 2009.01.19, Monday;

EXAMPLE 8 year = 0x07D9, month = 0x01, dayOfMonth = 0x13, dayofWeek = 0x02: error, as the dayOfMonth and dayOfWeek in the given year and month do not match.

time OCTET STRING (SIZE(4))
{
hour,
minute,
second,
hundredths
}
hour: interpreted as unsigned
range 0…23, 0xFF,
minute: interpreted as unsigned
range 0…59, 0xFF
second: interpreted as unsigned
range 0…59, 0xFF,
hundredths: interpreted as unsigned
range 0…99, 0xFF

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

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

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

clock_status unsigned interpreted as 8 bit string
The status bits are defined as follows:
bit 0 (LSB): invalid [a] value,
bit 1: doubtful [b] value,
bit 2: different clock base [c],
bit 3: invalid clock status [d],
bit 4: reserved,
bit 5: reserved,
bit 6: reserved,
bit 7 (MSB): daylight saving active [e]

date_time OCTET STRING (SIZE(12))
{
year highbyte,
year lowbyte,
month,
day of month,
day of week,
hour,
minute,
second,
hundredths of second,
deviation highbyte,
deviation lowbyte,
clock status
}

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

[a] Time could not be recovered after an incident. Detailed conditions are manufacturer specific (for example after the power to the clock has been interrupted).

[b] Time could be recovered after an incident but the value cannot be guaranteed. Detailed conditions are manufacturer specific.

[c] Bit is set if the basic timing information for the clock at the actual moment is taken from a timing source different from the source specified in clock_base.

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

[e] Flag set to true: the transmitted time contains the daylight saving deviation (summer time).

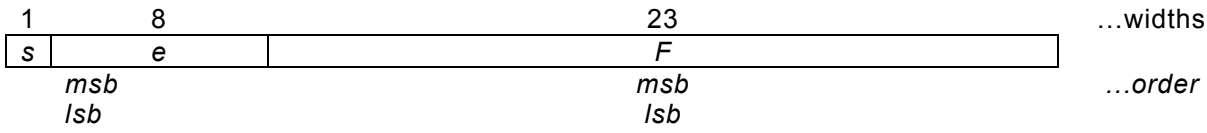
Flag set to false: the transmitted time does not contain daylight saving deviation (normal time).

4.6.2 Floating point number formats

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

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

The single format is:



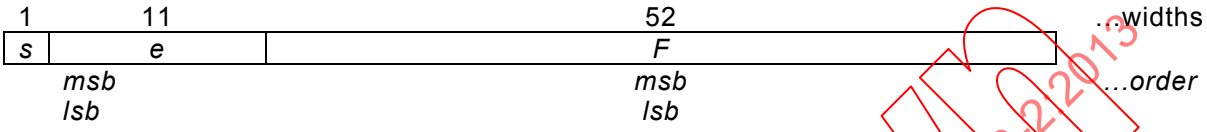
where:

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

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

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

The double format is:



where:

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

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

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

For more details, see ISO/IEC/IEEE 60559.

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

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

Bit 31 Sign bit **0** 0: + 1: -	Bits 30-23 Exponent field: **01111111** Decimal value of exponent field and exponent: **127**-127 = 0	Bits 22-0 Significand 1,**00000000000000000000000** Decimal value of the significand: 1,0000000

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

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

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

Bit 63 Sign bit **0** 0: + 1: -	Bits 62-52 Exponent field: **01111111111** Decimal value of exponent field and exponent: **1023**-1023 = 0	Bits 51-0 Significand 1,**0000000000000000000000000000000000000000000000000000** Decimal value of the significand: 1,0000000000000000

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

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

Bit 31 Sign bit **0** 0: + 1: -	Bits 30-23 Exponent field: **10001110** Decimal value of exponent field and exponent: **142**-127 = 15	Bits 22-0 Significand 1,**11100100110100000000000** Decimal value of the significand: 1,8937988

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

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

Bit 63 Sign bit **0** 0: + 1: -	Bits 62-52 Exponent field: **10000001110** Decimal value of exponent field and exponent: **1038**-1023 = 15	Bits 51-0 Significand 1,**1110010011010000000000000000000000000000000000000000** Decimal value of the significand: 1,8937988281250000

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

4.7 The COSEM server model

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

Level 1: Physical device;

Level 2: Logical device;

Level 3: Accessible COSEM objects.

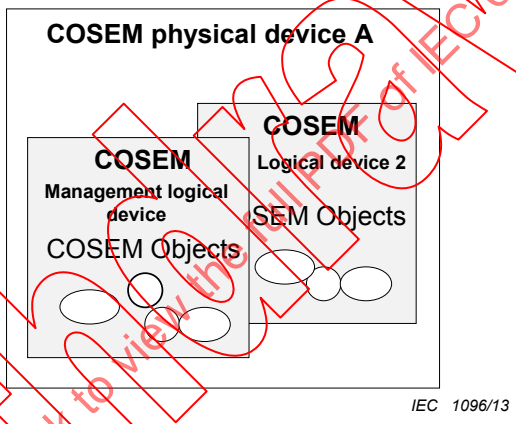


Figure 2 – The COSEM server model

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

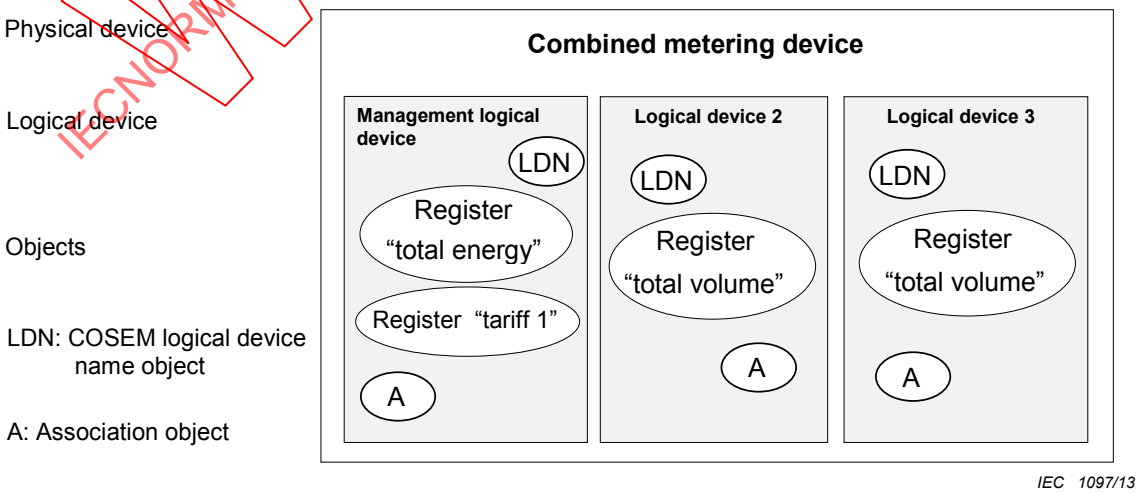


Figure 3 – Combined metering device

4.8 The COSEM logical device

4.8.1 General

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

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

4.8.2 COSEM logical device name

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

The logical device name is defined as an octet-string of up to 16 octets. The first three octets shall carry the manufacturer identifier [5]. The manufacturer shall ensure that the logical device name, starting with the three octets identifying the manufacturer and followed by up to 13 octets, is unique.

4.8.3 The “association view” of the logical device

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

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

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

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

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

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

4.8.4 Mandatory contents of a COSEM logical device

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

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

---

5 Administered by the DLMS User Association.

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

4.8.5 Management logical device

As specified in 4.8.1, the management logical device is a mandatory element of any physical device. It has a reserved address. It shall support an AA to a public client with the lowest security level. Its role is to support revealing the internal structure of the physical device and to support notification of events in the server.

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

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

4.9 Data security

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

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

For a description of these security mechanisms, see IEC 62056-5-3:—, Clause 5.

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

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

5 The COSEM interface classes

5.1 Overview

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

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

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

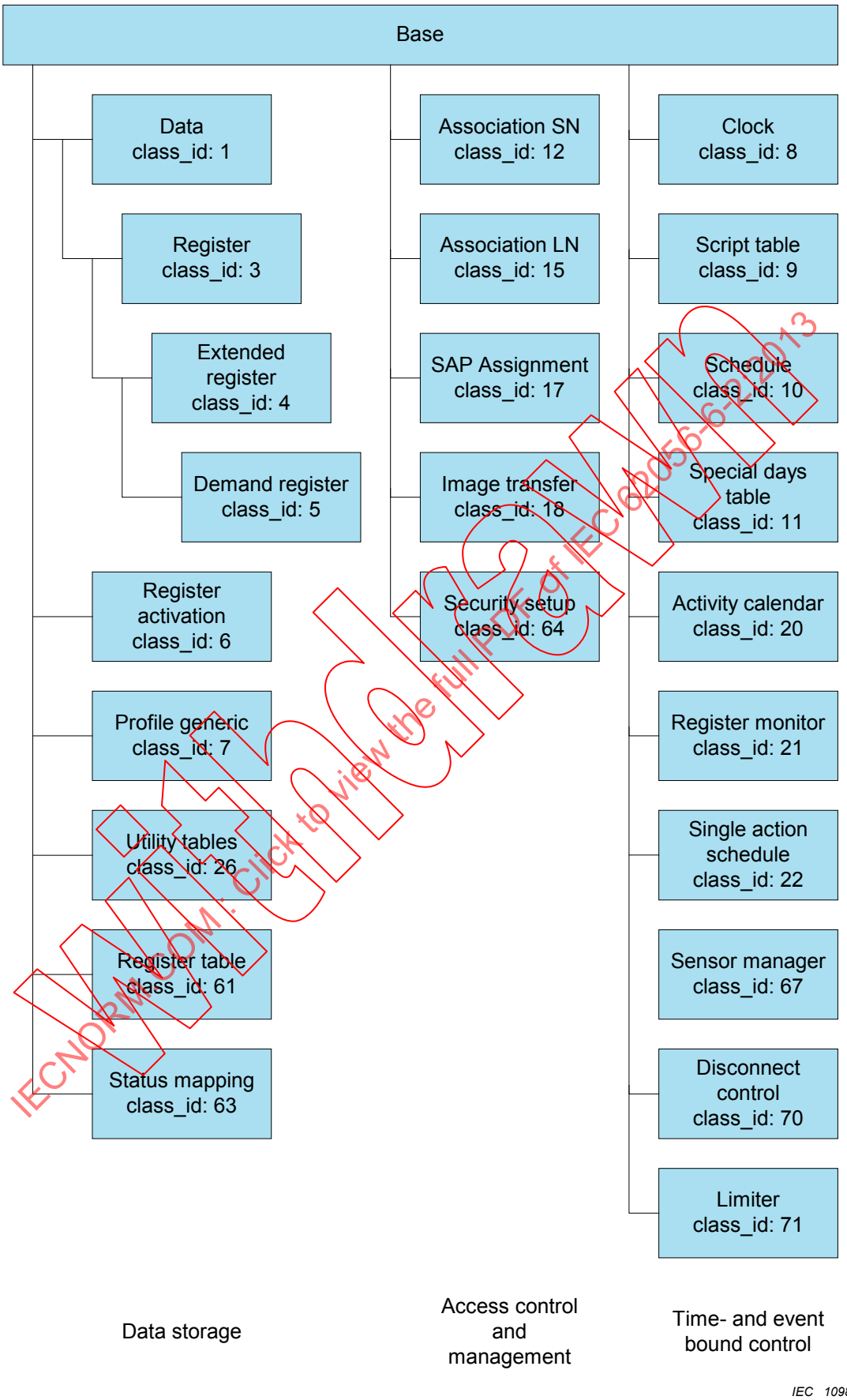


Figure 4 – Overview of the interface classes – Part 1

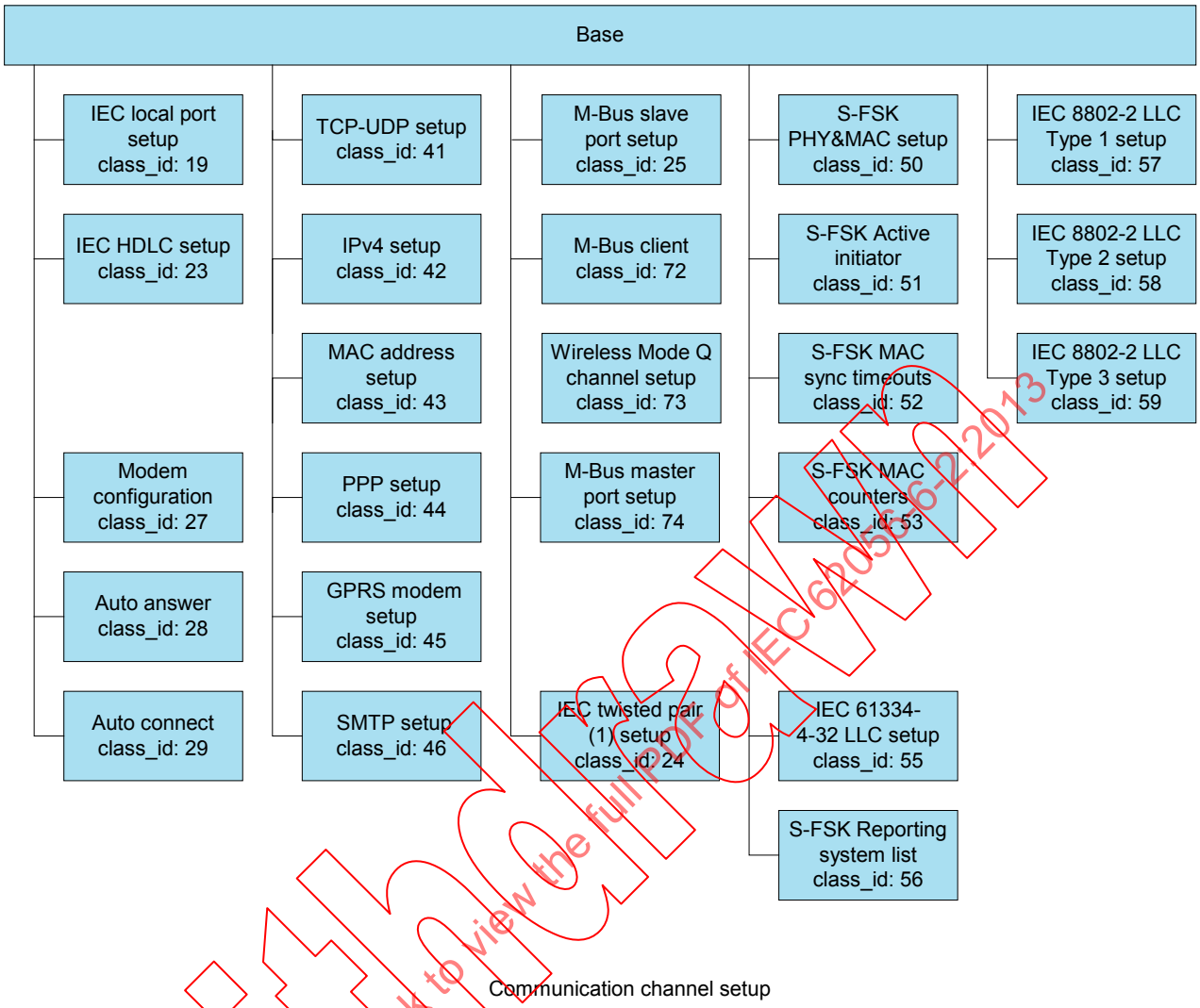


Figure 5 – Overview of the interface classes – Part 2

5.2 Interface classes for parameters and measurement data

5.2.1 Data (class_id: 1, version: 0)

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

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

Attribute description

logical_name	Identifies the “Data” object instance. See Clause 6 and IEC 62056-6-1.
value	Contains the data.

```
CHOICE
{
    -- simple data types
    null-data                [0],
    boolean                  [3],
    bit-string               [4],
    double-long              [5],
    double-long-unsigned     [6],
    octet-string             [9],
    visible-string           [10],
    UTF8-string              [12],
    bcd                      [13],
    integer                  [15],
    long                     [16],
    unsigned                 [17],
    long-unsigned            [18],
    long64                   [20],
    long64-unsigned          [21],
    enum                     [22],
    float32                  [23],
    float64                  [24],
    date-time                [25],
    date                     [26],
    time                     [27],
    -- complex data types
    array                    [1],
    structure                [2],
    compact-array            [19]
}
```

The data type depends on the instantiation defined by the “logical name” and possibly from the manufacturer. It has to be chosen so, that together with the logical name, an unambiguous interpretation is possible. Any simple and complex data types listed in 4.5 can be used, unless the choice is restricted in Clause 6.

5.2.2 Register (class_id: 3, version: 0)

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

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

Attribute description

logical_name	Identifies the “Register” object instance. See Clause 6 and IEC 62056-6-1.
value	Contains the current process or status value.

	CHOICE
	{
	-- simple data types
	null-data [0],
	bit-string [4],
	double-long [5],
	double-long-unsigned [6],
	octet-string [9],
	visible-string [10],
	UTF8-string [12],
	integer [15],
	long [16],
	unsigned [17],
	long-unsigned [18],
	long64 [20],
	long64-unsigned [21],
	float32 [23],
	float64 [24]
	}
		The data type of the value depends on the instantiation defined by "logical_name" and possibly on the choice of the manufacturer. It has to be chosen so that, together with the logical_name, an unambiguous interpretation of the value is possible.
		
		When, instead of a "Data" object, a "Register" object is used, (with the scaler_unit attribute not used or with scaler = 0, unit = 255) then the data types allowed for the value attribute of the "Data" IC are allowed.

scaler_unit	Provides information on the unit and the scaler of the value.
	
	scal_unit_type::= structure
	{
	scaler,
	unit
	}
	scaler: integer
	
	This is the exponent (to the base of 10) of the multiplication factor.
	
	REMARK If the value is not numerical, then the scaler shall be set to 0.
	
	unit: enum
	
	Enumeration defining the physical unit; for details, see Table 3 below.	

Method description

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

Table 3 – Enumerated values for physical units

unit:: = enum	Unit	Quantity	Unit name	SI definition (comment)
(1)	a	time	year	
(2)	mo	time	month	
(3)	wk	time	week	7*24*60*60 s
(4)	d	time	day	24*60*60 s
(5)	h	time	hour	60*60 s
(6)	min.	time	min	60 s
(7)	s	time (t)	second	s
(8)	°	(phase) angle	degree	rad*180/π
(9)	°C	temperature (T)	degree-celsius	K-273.15
(10)	currency	(local) currency		
(11)	m	length (l)	metre	m
(12)	m/s	speed (v)	metre per second	m/s
(13)	m^3	volume (V) r_V , meter constant or pulse value (volume)	cubic metre	m^3
(14)	m^3	corrected volume	cubic metre	m^3
(15)	m^3/h	volume flux	cubic metre per hour	$m^3/(60*60s)$
(16)	m^3/h	corrected volume flux	cubic metre per hour	$m^3/(60*60s)$
(17)	m^3/d	volume flux		$m^3/(24*60*60s)$
(18)	m^3/d	corrected volume flux		$m^3/(24*60*60s)$
(19)	l	volume	litre	10^{-3} m^3
(20)	kg	mass (m)	kilogram	
(21)	N	force (F)	newton	
(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	joule per hour	J/(60*60s)
(27)	W	active power (P)	watt	W = J/s
(28)	VA	apparent power (S)	volt-ampere	
(29)	var	reactive power (Q)	var	
(30)	Wh	active energy r_W , active energy meter constant or pulse value	watt-hour	W*(60*60s)
(31)	VAh	apparent energy r_S , apparent energy meter constant or pulse value	volt-ampere-hour	VA*(60*60s)
(32)	varh	reactive energy r_B , reactive energy meter constant or pulse value	var-hour	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	electric field strength (E)	volt per metre	V/m
(37)	F	capacitance (C)	farad	C/V = As/V
(38)	Ω	resistance (R)	ohm	Ω = V/A
(39)	$Ωm^2/m$	resistivity (ρ)		Ωm
(40)	Wb	magnetic flux (Φ)	weber	Wb = Vs
(41)	T	magnetic flux density (B)	tesla	Wb/m^2
(42)	A/m	magnetic field strength (H)	ampere per metre	A/m

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

Some examples are shown in Table 4 below.

Table 4 – Examples for scaler_unit

Value	Scaler	Unit	Data
263788	-3	m^3	263,788 m^3
593	3	Wh	593 kWh
3467	-1	V	346,7
3467	0	V	3 467 V
3467	1	V	34 670 V

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

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

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

Attribute description

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

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

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

```
CHOICE
{
    -- simple data types
    null-data                [0],
    bit-string               [4],
    double-long-unsigned     [6],
    octet-string             [9],
    visible-string           [10],
    UTF8-string              [12],
    unsigned                 [17],
    long-unsigned            [18],
    long64-unsigned          [21]
}
```

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

Def. Depending on the status type definition.

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

	octet-string, formatted as set in 4.6.1 for *date_time*

Method description

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

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

This IC allows modelling a demand value, with its associated scaler, unit, status and time information. A "Demand register" object measures and computes a *current_average_value* periodically, and it stores a *last_average_value.* The time interval *T* over which the demand is measured or computed is defined by specifying "*number_of_periods*" and "*period*". See Figure 6.

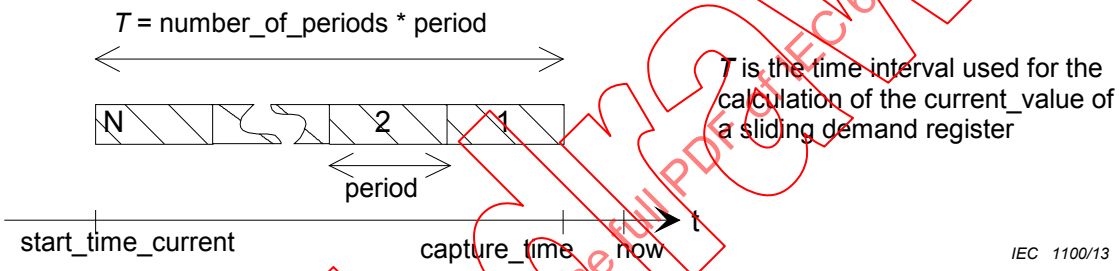


Figure 6 – The time attributes when measuring sliding demand

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

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

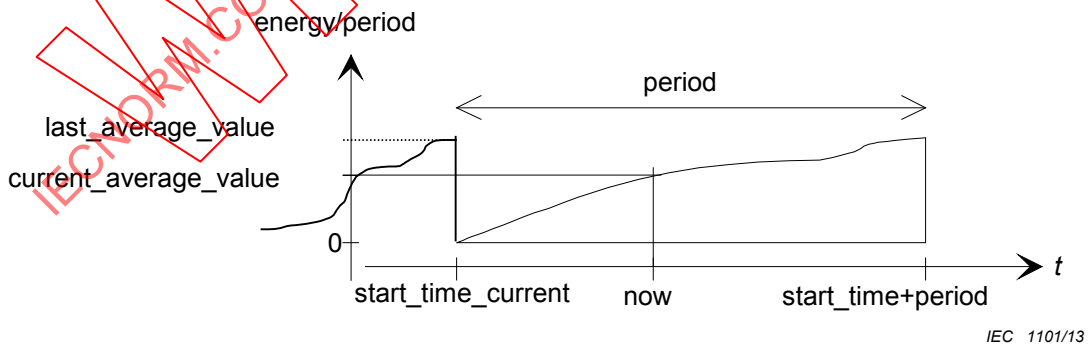


Figure 7 – The attributes in the case of block demand

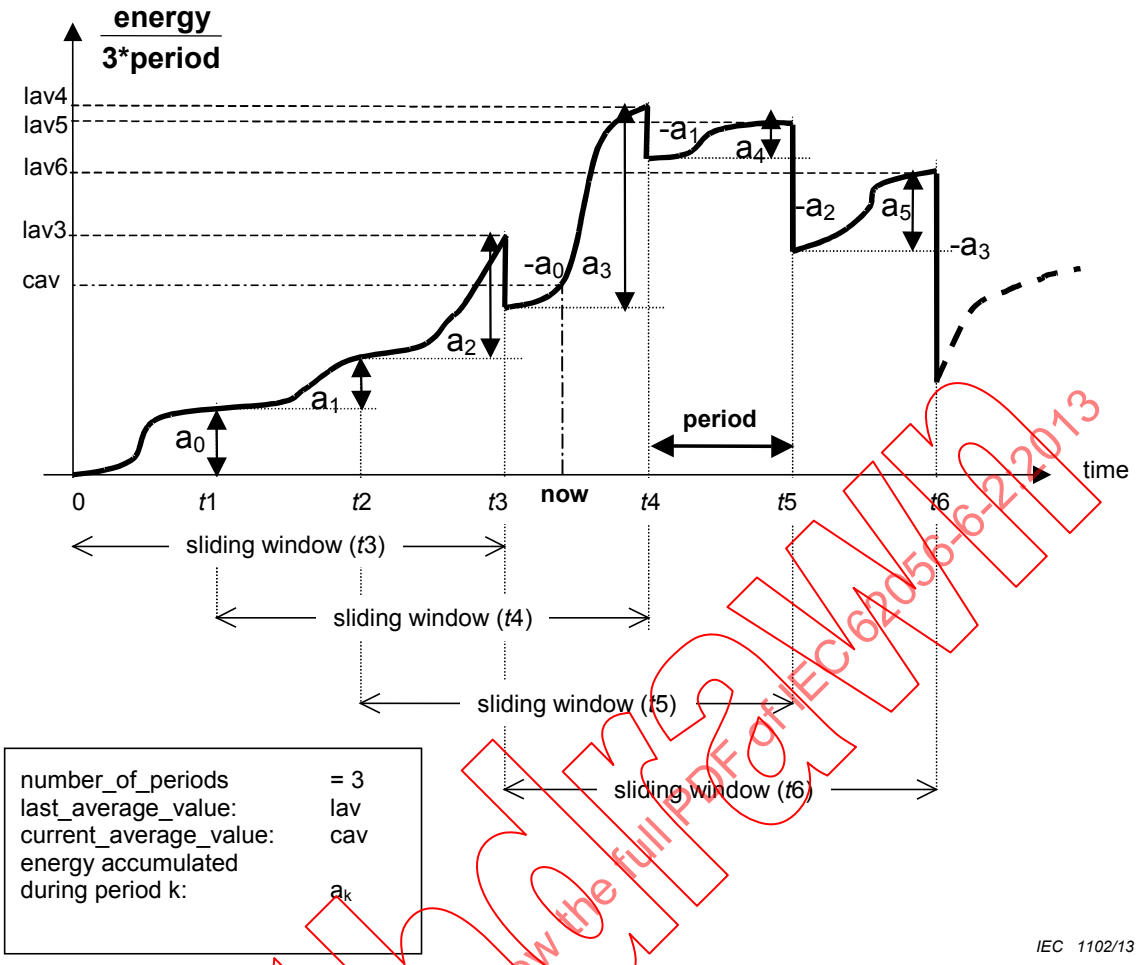


Figure 8 – The attributes in the case of sliding demand (number of periods = 3)

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

Attribute description

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

logical_name	Identifies the “Demand register” object instance. See 6.3.1.
current_average_value	Provides the current value (running demand) of the energy accumulated since start_time, divided by number_of_periods*period. The data type of the value depends on the instantiation defined by logical_name and possibly on the choice of the manufacturer. The type has to be chosen so that, together with the logical_name, unambiguous interpretation of the value is possible. If a quantity other than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current). CHOICE — For data types, see “Register” IC, value attribute.
last_average_value	Provides the value of the energy accumulated (over the last number_of_periods*period), divided by number_of_periods*period. The energy of the current (not terminated) period is not considered by the calculation. If a quantity other than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current). CHOICE — For data types, see “Register” IC, value attribute.
status	Provides “Demand register” specific status information. The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary. CHOICE — For data types, see “Extended register” IC, status attribute. ***Def.*** — Depending on the status type definition.
capture_time	Provides the date and time when the last_average_value has been calculated. octet-string, formatted as set in 4.6.1 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 4.6.1 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 number_of_periods > 1 indicates that the last_average_value represents “sliding demand”. number_of_periods = 1 indicates that the last_average_value represents "block demand". The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.

Method description

reset (data)	This method forces a reset of the object. Activating this method provokes the following actions: • the current period is terminated; • the current_average_value and the last_average_value are set to their default values; • the capture_time and the start_time_current are set to the time of the execution of *reset (data)*. data::= integer(0)
next_period (data)	This method is used to trigger the regular termination (and restart) of a period. Closes (terminates) the current measuring period. Updates capture_time and start_time and copies current_average_value to last_average_value, sets current_average_value to its default value. Starts the next measuring period. REMARK The old last_average_value (and capture_time) can be read during the time “period”. The old current_average_value is not available any more at the interface. data::= integer(0)

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

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

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

Attribute description

logical_name

Identifies the “Register activation” object instance. See 6.2.10.

register_assignment

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

array object_definition

object_definition::= structure

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

mask_list

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

array register_act_mask

register_act_mask::= structure

```
{
    mask_name:      octet-string,
    index_list:     index_array
}
```

mask_name has to be uniquely defined within the object

index_array::= array unsigned

active_mask

Defines the currently active mask. The mask is defined by its mask_name (see mask_list).

The active_mask defines the registers currently enabled; all other registers listed in the register_assignment are disabled.

Method description

add_register (data)	Adds one more registers to the attribute register_assignment. The new register is added at the end of the array; i.e. the newly added register has the highest index. The indices of the existing registers are not modified.
	
	data::= structure
	
	{
	class_id: long-unsigned,
	logical_name: octet-string
	}

add_mask (data)	Adds another mask to the attribute mask_list. If there already exists a mask with the same name, the existing mask will be overwritten by the new mask. data::= register_act_mask (see above)
delete_mask (data)	Deletes a mask from the attribute mask_list. The mask is defined by its mask name. data::= octet-string (mask_name)

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

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

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

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

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

The size of profile data is determined by three parameters:

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

b) the maximum number of entries to retain (*profile_entries*). If all entries are filled and a capture () request occurs, the least important entry (according to the requested sorting method) will get lost. This maximum number of entries may be specified. Upon changing it, the buffer will be adjusted;

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

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

Attribute description

logical_name

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

buffer

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

compact-array or array entry

entry::= structure

```
{
    CHOICE
    {
        -- simple data types
        null-data               [0],
        boolean                 [3],
        bit-string              [4],
        double-long             [5],
        double-long-unsigned    [6],
        octet-string            [9],
        visible-string          [10],
        UTF8-string             [12],
        bcd                     [13],
        integer                 [15],
        long                    [16],
        unsigned                [17],
        long-unsigned           [18],
        long64                  [20],
        long64-unsigned         [21],
        enum                    [22],
        float32                 [23],
        float64                 [24],
```

```
                date-time                [25],
                date                     [26],
                time                     [27],
                -- complex data types
                array                    [1],
                structure                [2],
                compact-array            [19]
            }
}
```

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

Default: The buffer is empty after reset.

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

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

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

---

capture_objects

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

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

array capture_object_definition

capture_object_definition::= structure

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

- where attribute_index is a pointer to the attribute within the object. attribute_index 1 refers to the 1st attribute (i.e. the logical_name), attribute_index 2 to the 2nd, etc.); attribute_index 0 refers to all public attributes;

- where data_index is a pointer selecting a specific element of the attribute. The first element in the attribute structure is identified by data_index 1. If the attribute is not a structure, then the data_index has no meaning. If the capture object is the buffer of a profile, then the data_index identifies the captured object of the buffer (i.e. the column) of the inner profile.

- data_index 0: references the whole attribute

---

capture_period

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

	0: No automatic capturing; capturing is triggered externally or capture events occur asynchronously.
sort_method	If the profile is unsorted, it works as a “first in first out” buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the Clock object). If the buffer is full, the next call to capture () will push out the first (oldest) entry of the buffer to make space for the new entry. If the profile is sorted, a call to capture () will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the buffer after the last entry and if the buffer is already full, the new entry will not be retained at all. enum: (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) NOTE 1 If the sort_method is FIFO or LIFO, then all elements of the capture_object_definition specifying the sort object can be zero.
entries_in_use	Counts the number of entries stored in the buffer. After a call of the reset () method, the buffer does not contain any entries, and this value is zero. Upon each subsequent call of the capture () method, this value will be incremented up to the maximum number of entries that will get stored (see profile_entries). double-long-unsigned: 0…profile_entries ***Def.*** 0
profile_entries	Specifies how many entries shall be retained in the buffer. double-long-unsigned: 1…(limited by physical size) ***Def.*** 1

Parameters for selective access to the buffer attribute

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

range_descriptor::= structure

{

restricting_object: capture_object_definition		Defines the capture_object restricting the range of entries to be retrieved. Only simple data types are allowed.
from_value:		Oldest or smallest entry to retrieve
	CHOICE	
	{-- simple data types	
	double-long [5],	
	double-long-unsigned [6],	
	octet-string [9],	
	visible-string [10],	
	UTF8-string [12],	
	integer [15],	
	long [16],	
	unsigned [17],	
	long-unsigned [18],	
	long64 [20],	
	long64-unsigned [21],	
	float32 [23],	
	float64 [24],	
	date-time [25],	
	date [26],	
	time [27]	
	}	
to_value:	CHOICE	Newest or largest entry to retrieve
	{see above}	
selected_values:	array capture_object_definition	
	List of columns to retrieve. If the array is empty (has no entries), all captured data are returned. Otherwise, only the columns specified in the array are returned. The type *capture_object_definition* is specified above (*capture_objects*).	

}

entry_descriptor::= structure

{

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

}

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

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

Method description

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

Behaviour of the object after modification of certain attributes

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

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

Restrictions

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

Profile used to define a subset of preferred readout values

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

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

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

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

Utility tables	0...n	class_id = 26, version = 0			
Attributes	*Data type*	*Min.*	*Max.*	*Def.*	Short name
Specific methods	*m/o*				

Attribute description

logical_name	Identifies the “Utility tables” object instance. See 6.2.27.
table_ID	Table number. This table number is as specified in the ANSI standard and may be either a standard table or a manufacturer’s table.
length	Number of octets in table buffer.
buffer	Contents of the table. **Selective access** (see 4.4) to the attribute *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 sub-tree of the hierarchy starting at the selection point.

}

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

This IC allows to group homogenous entries, identical attributes of multiple objects, which are all instances of the same IC, and in their logical name (OBIS code) the value in value groups A to D and F is identical. The possible values in value group E are defined in IEC 62056-6-1 in a tabular form: the table header defines the common part of the OBIS code and each table cell defines one possible value of value group E. A “Register table” object may capture attributes of some or all of those objects.

NOTE 1 Some examples are the “UNIPEDE voltage dip quantities” table, see 1IEC 62056-6-1:—, Table 18, or the “Extended phase angle measurement” table, see IEC 62056-6-1:—, Table 16.

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

Register table		0...n	class_id = 61, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. table_cell_values	(dyn.)	Array				x + 0x08
3. table_cell_definition	(static)	Structure				x + 0x10
4. scaler_unit	(static)	scaler_unit_type				x + 0x18
Specific methods		***m/o***				
1. reset (data)		O				x + 0x28
2. capture (data)		O				x + 0x30

Attribute description

logical_name

Identifies the “Register table” object instance.

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

When the format of the logical name is A.B.98.10.x.255, several instances of the “Register table” IC can be used to capture different attributes of the objects concerned. The value group E numbers the instances. See IEC 62056-6-1:—, 6.4.

table_cell_values

Holds the value of the attributes captured, as they would be returned to a GET or Read .request to the individual attributes.

compact array or array table_cell_entry

table_cell_entry

CHOICE

{
-- simple data types
null-data [0],
bit-string [4],
double-long [5],
double-long-unsigned [6],
octet-string [9],
visible-string [10],
UTF8-string [12],
bcd [13],
integer [15],
long [16],
unsigned [17],
long-unsigned [18],
long64 [20],
long64-unsigned [21],
float32 [23],
float64 [24],
-- complex data types
structure [2]

}

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

table_cell_definition

Specifies the list of attributes captured in the register table.

structure

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

where:

- class_id defines the common class_id of the objects the attributes of which are captured;
- logical_name contains the common logical name of the objects, with E = 255 (wildcard);
- group_E_values contain the list of cell identifiers, as defined in the respective table of IEC 62056-6-1;
- attribute_index is a pointer to the attribute within the object. attribute_index 0 refers to all public attributes.

If the logical name of the “Register table” object is in the format A.B.C.D.255.F and the defined attribute of all objects identified in the respective table in IEC 62056-6-1 are captured, then attribute 3 may not be accessible. In this case:

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

scaler_unit

See the description of IC “Register”.

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

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

Method description

reset (data)	Clears the table_cell_values. It has no effect on the attributes captured. data::= integer(0)
capture (data)	Copies the values of the attributes into the table_cell_values. If the attribute_index = 0, all attributes are captured.

Behaviour of the object after modification of the table_cell_definition attribute

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

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

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

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

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

Attribute description

logical_name	Identifies the "Status mapping" object instances. See 6.2.31, 6.2.33, 6.2.39 and 6.3.7.

status_word	Contains the current value of the status word.
	
	CHOICE
	
	{
	bit-string [4],
	double-long-unsigned [6],
	octet-string [9],
	visible-string, [10],
	UTF8-string [12],
	unsigned [17],
	long-unsigned [18],
	long64-unsigned [21]
	}
	
	The size of the status_word is n*8 bits, the maximum size is 65 536 bits.
	
	NOTE 1 Manufacturers can choose any of the types listed above. However, the status word is always interpreted as a bit-string.

mapping_table

Contains the mapping of the status word to the positions in the reference table.

structure

```
{       ref_table_id:       unsigned,
        CHOICE
        {
                first_entry:  long-unsigned,
                array:        table_entries
        }
}
        table_entry:        long-unsigned
```

where:

- ref_table_id is the identifier of the reference status table

  NOTE 2 Reference status tables are maintained by the DLMS UA. They can also be project specific.

- first_entry is the entry in the reference table corresponding to bit 0 of the status word. Bit 1 corresponds to the next entry and so on, the last entry is defined by the length of the status word;
- if the “array” choice is taken, the array of table entries maps the bits of the status word to the entries in the reference status table. The position in the array defines the position in the status word (the first position corresponds to bit 0).

5.3 Interface classes for access control and management

5.3.1 Association SN (class_id: 12, version: 2)

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

The **short_name** of the “Association SN” object itself is fixed within the COSEM context. See 4.3.

Association SN		0...n	class_id = 12, version = 2			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. object_list	(static)	objlist_type				x + 0x08
3. access_rights_list	(static)	access_rights_type				x + 0x10
4. security_setup_reference	(static)	octet-string				x + 0x18
Specific methods		***m/o***				
1. *reserved from previous versions*		o				
2. *reserved from previous versions*		o				
3. read_by_logicalname (data)		o				x + 0x30
4. *reserved from previous versions*		o				
5. change_secret (data)		o				x + 0x40
6. *reserved from previous versions*		o				
7. *reserved from previous versions*						
8. reply_to_HLS_authentication (data)		o				x + 0x58

Attribute description

logical_name

Identifies the "Association SN" object instance. See 6.2.22.

object_list

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

object_list_type::= array objlist_element

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

selective access (see 4.4) to the attribute object_list may be available (optional). The access selector values and their parameters are as defined below.

access_rights_list

Contains the access rights to attributes and methods.

The link between the object_list and the access_rights_list is the base_name, present in both the object_list_element structure and the access_right_element structure. Therefore, the base_names on the two lists shall be the same.

The number – and preferably, the order – of the elements in the array of object_list_element and the array of access_right_element shall also be the same.

access_rights_type::= array access_rights_element

```
access_rights_element::= structure

{
        base_name:              long,
        attribute_access: attribute_access_descriptor,
        method_access:  method_access_descriptor
}
attribute_access_descriptor::= array            attribute_access_item

attribute_access_item::= structure
{
        attribute_id:           integer,
        access_mode:            enum,
                                {
                                    (0)    no_access,
                                    (1)    read_only,
                                    (2)    write_only,
                                    (3)    read_and_write,
                                    (4)    authenticated_read_only,
                                    (5)    authenticated_write_only,
                                    (6)    authenticated_read_and_write
                                }
        access_selectors:       CHOICE
                                {
                                    null-data,
                                    array of integer
                                }
}
method_access_descriptor::= array       method_access_item

method_access_item::= structure
{
        method_id: integer,
        access_mode:            enum
                        {
                                    (0)     no_access,
                                    (1)     access,
                                    (2)     authenticated_access
                        }
}
```

selective access (see 4.4) to the attribute access_rights_list may be available (optional). The access selector values and their parameters are as defined below.

security_setup_ reference	References the Security setup object by its logical name. The referenced object manages security for a given Association SN object instance.

Parameters for selective access to the object_list and access_right_list attribute

Access selector value	Parameter	Available with attribute	Comment
1	class_id: long-unsigned	2	Delivers the subset of the object_list for a specific class_id. For the response: data::= object_list_type

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

3	base_name: long	2, 3	In the case of attribute 2, delivers the entry of the object_list for a specific base_name. For the response: data::= object_list_element In the case of attribute 3, delivers the entry of the access_rights_list for a specific base_name. For the response: data::=access_rights_element

Method description

read_by_ logicalname (data)

Reads attributes for selected objects. The objects are specified by their class_id and their logical_name. With this method, the parameterised access feature can also be used.

data::= array attribute_identification

attribute_identification::= structure

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

where attribute_index is a pointer (i.e. offset) to the attribute within the object.

attribute_index 0 delivers all attributes [a], attribute_index 1 delivers the first attribute (i.e. logical_name, etc.).

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

change_secret (data)

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

data::= octet-string new secret[b]

NOTE In the case of HLS with GMAC, the (HLS_)secret is held by the Security setup object referenced in attribute 4.

reply_to_HLS_ authentication (data)

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

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

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

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

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

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

[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 violated", see IEC 62056-5-3:—,Clause 8.

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

5.3.2 Association LN (class_id: 15, version: 1)

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

Association LN	**0...MaxNbofAss.**	**class_id = 15, version = 1**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. object_list (static)	object_list_type				x + 0x08
3. associated_partners_id	associated_partners_type				x + 0x10
4. application_context_name	application_context_name				x + 0x18
5. xDLMS_context_info	xDLMS_context_type				x + 0x20
6. authentication_mechanism_name	mechanism_name				x + 0x28
7. secret	octet-string				x + 0x30
8. association_status	enum				x + 0x38
9. security_setup_reference (static)	octet-string				x + 0x40
Specific methods	***m/o***				
1. reply_to_HLS_authentication (data)	o				x + 0x60
2. change_HLS_secret (data)	o				x + 0x68
3. add_object (data)	o				x + 0x70
4. remove_object (data)	o				x + 0x78

Attribute description

logical_name	Identifies the "Association LN" object instance. See 6.2.22.

object_list

Contains the list of visible COSEM objects with their class_id, version, logical name and the access rights to their attributes and methods within the given application association.

```
object_list_type::= array          object_list_element

object_list_element::= structure
{
        class_id:                  long-unsigned,
        version:                   unsigned,
        logical_name:              octet-string,
        access_rights:             access_right
}
access_right::= structure
{
        attribute_access:          attribute_access_descriptor,
        method_access:             method_access_descriptor
}
attribute_access_descriptor::= array       attribute_access_item

attribute_access_item::=  structure
{
        attribute_id:              integer,
        access_mode:               enum
                                   {
                                       (0)    no_access,
                                       (1)    read_only,
                                       (2)    write_only,
                                       (3)    read_and_write,
                                       (4)    authenticated_read_only,
                                       (5)    authenticated_write_only,
                                       (6)    authenticated_read_and_write
                                   }
        access_selectors:          CHOICE
                                   {
                                       null-data,
                                       array of integer
                                   }
}
method_access_descriptor::= arraymethod_access_item

method_access_item::= structure
{
        method_id: integer,
        access_mode:     enum
                         {
                             (0)    no_access,
                             (1)    access,
                             (2)    authenticated_access
                         }
}
```

where:

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

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

associated_partners_id

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

associated_partners_type::= structure

{
client_SAP: integer,
server_SAP: long-unsigned
}

The range for the client_SAP is 0…0x7F.

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

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

---

application_context_name

In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that association.

CHOICE
{
structure [2],
octet-string [9]
}

The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:—, 7.2.2.2. The application_context_name attribute includes the arc labels of the OBJECT IDENTIFIER. In this case:

application_context_name::= structure

{
joint-iso-ctt-element: unsigned,
country-element: unsigned,
country-name-element: long-unsigned,
identified-organization-element: unsigned,
DLMS-UA-element: unsigned,
application-context-element: unsigned,
context-id-element: unsigned
}

For existing implementations, the attribute may hold the value of the OBJECT IDENTIFIER encoded in BER, as an octet string. See IEC 62056-5-3:—, B.4. In this case:

application_context_name::= octet-string

// holds the value of the OBJECT identifier encoded in BER

Examples:

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

The A-XDR encoding as an octet-string, holding the value of the OBJECT IDENTIFIER encoded in BER (all values are hexadecimal): 09 07 60 85 74 05 08 01 01.

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

}

where:

- the conformance element contains the xDLMS conformance block supported by the server;
- the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS InitiateResponse APDU;
- the max_send_pdu_size, in an active AA, contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the xDLMS InitiateRequest APDU;
- the dlms_version_number element contains the DLMS version number supported by the server;
- the quality_of_service element is not used;
- the cyphering_info, in an active association, contains the dedicated key parameter of the xDLMS InitiateRequest APDU. See IEC 62056-5-3:—, Clause 8.

authentication_ mechanism_name	Contains the name of the authentication mechanism for the association.
	
	Mechanism-name CHOICE
	{
	structure [2],
	octet-string [9]
	}
	The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-3:—, 7.2.2.3.
	
	The authentication_mechanism_name attribute includes the arc labels of the OBJECT IDENTIFIER. In this case:
	
	authentication_mechanism_name::= structure
	
	{
	joint-iso-ctt-element: unsigned,
	country-element: unsigned,
	country-name-element: long-unsigned,
	identified-organization-element: unsigned,
	DLMS-UA-element: unsigned,
	authentication-mechanism-name-element: unsigned,
	mechanism-id-element: unsigned
	}
	For existing implementations, the attribute may hold the value of the OBJECT IDENTIFIER encoded in BER, as an octet string. See IEC 62056-5-3:—, B.4. In this case:
	
	authentication_mechanism_name::= octet-string
	
	// holds the value of the OBJECT identifier encoded in BER
	
	Examples
	
	In the case of mechanism_id(1) the A-XDR encoding as a structure (all values are hexadecimal): 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01.
	
	The A-XDR encoding as an octet-string, holding the value of the OBJECT IDENTIFIER encoded in BER (all values are hexadecimal): 09 07 60 85 74 05 08 02 01.
	
	No mechanism-name is required when no authentication is used.

secret	Contains the secret for the LLS or HLS authentication process. NOTE In the case of HLS with GMAC, the (HLS_)secret is held by the Security setup object referenced in attribute 9.
association_status	Indicates the current status of the association, which is modelled by the object.
	enum: (0) non-associated, (1) association-pending, (2) associated
security_setup_ reference	References the Security setup object by its logical name. The referenced object manages security for a given Association LN object instance.

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

Parameters for selective access to the object_list attribute

- If no selective access is requested, (no Access_Selection_Parameters parameter is present in the GET.request (.indication) service primitive for the object_list attribute) the corresponding .response (.confirmation) service shall contain all object_list_elements of the object_list attribute.

- When selective access is requested to the object_list attribute (the Access_Selection_Parameter parameter is present), the response shall contain a 'filtered' list of object_list_elements, as follows:

Access selector	Access parameter	Comment
1	NULL	All information excluding the access_rights shall be included in the response.
2	class_list	Access by class_id. In this case, only those object_list_elements of the object_list shall be included in the response, which have a class_id equal to one of the class_id-s of the class_list. No access_right information is included. class_list::= array class_id class_id: long-unsigned

3	object_id_list	Access by object. The full information record of object instances on the object_Id_list shall be returned.
		object_id_list::= array object_id
		object_id::= structure
		{
		class_id: long-unsigned,
		logical_name: octet-string
		}

4	object_id	The full information record of the required COSEM object instance shall be returned. object_id::= structure See above.

Method description

reply_to_HLS_ authentication (data)	The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, f(StoC), as the *data* service parameter of the ACTION.request primitive invoked. data::= octet-string client's response to the challenge If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, f(CtoS) in the *data* service parameter of the response service. data::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.
change_HLS_secret (data)	Changes the HLS secret (for example encryption key). data::= octet-string[a] new HLS secret

add_object (data)	Adds the referenced object to the object_list. data::= object_list_element (see above)
remove_object (data)	Removes the referenced object from the object_list. data::= object_list_element (see above)

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

5.3.3 SAP assignment (class_id: 17, version: 0)

This IC allows modelling the logical structure of physical devices, by providing information on the assignment of the logical devices to their SAP-s. See IEC 62056-5-3:—, Annex A.

SAP assignment	**0...1**	**class_id = 17, version = 0**			
Attribute(s)	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. SAP_assignment_list (static)	asslist_type			0	x + 0x08
Specific methods	***m/o***				
1. connect_logical_device (data)	o				x + 0x20

Attribute description

logical_name	Identifies the "SAP assignment" objects instance. See 6.2.23.

SAP_assignment_list	Contains the list of all logical devices and their SAP addresses within the physical device.
	
	asslist_tpye::= array asslist_element
	
	asslist_element::= structure
	{
	SAP: long-unsigned,
	logical_device_name: octet-string CHOICE
	{
	octet-string [9],
	visible-string [10],
	Unicode-UTF8 [12]
	}
	}
	REMARK: The actual addressing is performed by the supporting communication layers.

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

5.3.4 Image transfer (class_id: 18, version: 0)

5.3.4.1 General

Instances of the Image transfer IC model the mechanism of transferring binary files, called firmware Images to COSEM servers. The Image transfer takes place in several steps:

Step 1: The client gets the ImageBlockSize from each server individually;

Step 2: The client initiates the Image transfer process individually or using broadcast;

Step 3: The client transfers ImageBlocks to (a group of) server(s) individually or using broadcast;

Step 4: The client checks the completeness of the Image in each server individually and transfers any ImageBlocks not (yet) transferred;

Step 5: The Image is verified;

Step 6: Before activation, the Image is checked;

Step 7: The Image(s) is (are) activated.

5.3.4.2 Definitions related to the Image transfer process

5.3.4.2.1
Image
binary data of specified size, which may be transferred, verified and activated

Note 1 to entry: The Image data transferred and the Image(s) that will be activated are not necessarily identical.

5.3.4.2.2
ImageSize
size of the Image expressed in bytes

Note 1 to entry: The Image can be transferred in ImageBlocks.

5.3.4.2.3
ImageBlock
part of the Image of size ImageBlockSize

Note 1 to entry: Each block is identified by its ImageBlockNumber

5.3.4.2.4
ImageBlockSize
size of ImageBlock expressed in bytes

5.3.4.2.5
ImageBlockNumber
identifier of an ImageBlock

Note 1 to entry: ImageBlocks are numbered sequentially, starting from 0.

The meaning of the definitions above is illustrated in Figure 9.

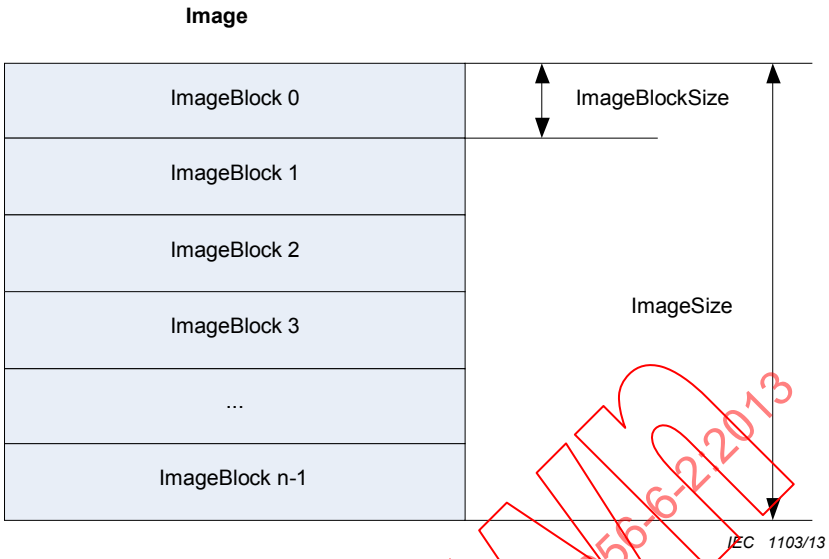


Figure 9 – The meaning of the definitions concerning the Image

5.3.4.3 The Image transfer mechanism

The Image transfer mechanism consists of two sets of COSEM client services; see Figure 10:

- the Image read services are used by the COSEM client to read ImageBlocks of the Image;

  NOTE 1 The generation of the Image, its location and the Image read services are out of the scope of this standard.

- the Image transfer services are used by the COSEM client to transfer the Image to the COSEM server.

The Image transfer services are mapped to COSEM services, accessing attributes and methods of (a) COSEM Image transfer interface object(s).

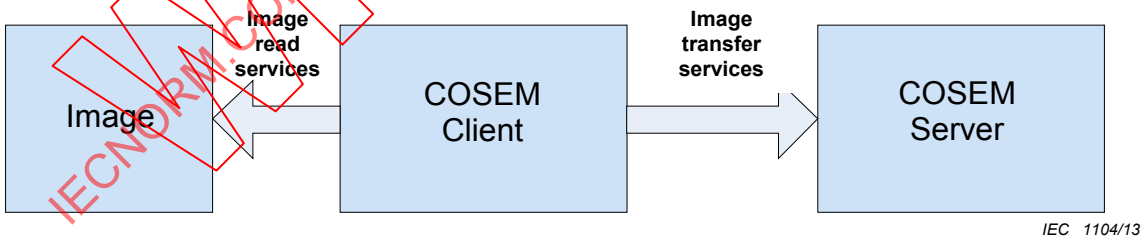


Figure 10 – The Image Read and the Image Transfer services

The Image transfer process can only be started if it is enabled in the server.

The process is shown in Figure 11 and it is explained below.

Step 1: Get ImageBlockSize: The first step is to get the ImageBlockSize supported by the server, so that the client can transfer Image blocks of the right size. The ImageBlockSize parameter is held by the *image_block_size* attribute and it may be different in different servers.

NOTE 2 The value of this attribute normally cannot be written by the client; it is a property of the server.

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

Step 2: Initiate Image transfer: In the second step, the COSEM client initiates Image transfer. Initiation is performed by invoking the *image_transfer_initiate* method. The method invocation parameter holds the identifier and the size of the Image to be transferred. The server shall make the memory space – necessary to accommodate the Image – available. Initiation of the Image transfer can be performed either individually or using broadcast.

After a successful initiation, the value of the *image_transfer_status* attribute is (1) *Image transfer initiated* and the COSEM server is prepared to accept ImageBlocks.

The value of the *image_to_activate_info* attribute shall be reset.

Step 3: Transfer ImageBlocks: In the third step, the client – after reading ImageBlocks of ImageBlockSize from the Image – transfers these ImageBlocks to the server(s). ImageBlocks may be transferred individually, or may be broadcast.

The transfer is performed by invoking the *image_block_transfer* method. The method invocation parameters include the ImageBlockNumber and one ImageBlock. ImageBlocks are accepted only by those COSEM servers, in which the Image transfer process has been successfully initiated. Other servers silently discard any ImageBlocks received.

Step 4: Check completeness of the Image: In the fourth step, the client checks – with each server individually – the completeness of the Image transferred. If the Image is not complete, it transfers the ImageBlocks not (yet) transferred. This is an iterative process, continued until the whole Image is successfully transferred.

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

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

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

NOTE 3 The two strategies can be freely combined.

Step 5: Verify Image: In the fifth step, the Image is verified. This is done by invoking the *image_verify* method by the client. The result can be:

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

NOTE 4 For xDLMS services, Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:—, Clause 8.

The verification may be initiated also by the server.

The result of the Image verification may be checked by the client by retrieving the value of the *image_transfer_status* attribute.

NOTE 5 The conditions of verifying the Image are out of the scope of this standard.

Step 6: Check the Image before activation: In the sixth step, the client may check the information on the Image transferred before activating it.

This information is held by the *image_to_activate_info* attribute of the Image transfer object and it is generated as a result of the Image verification. For each Image to be activated, this attribute holds the parameters: {image_size, image_identification, image_signature}.

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

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

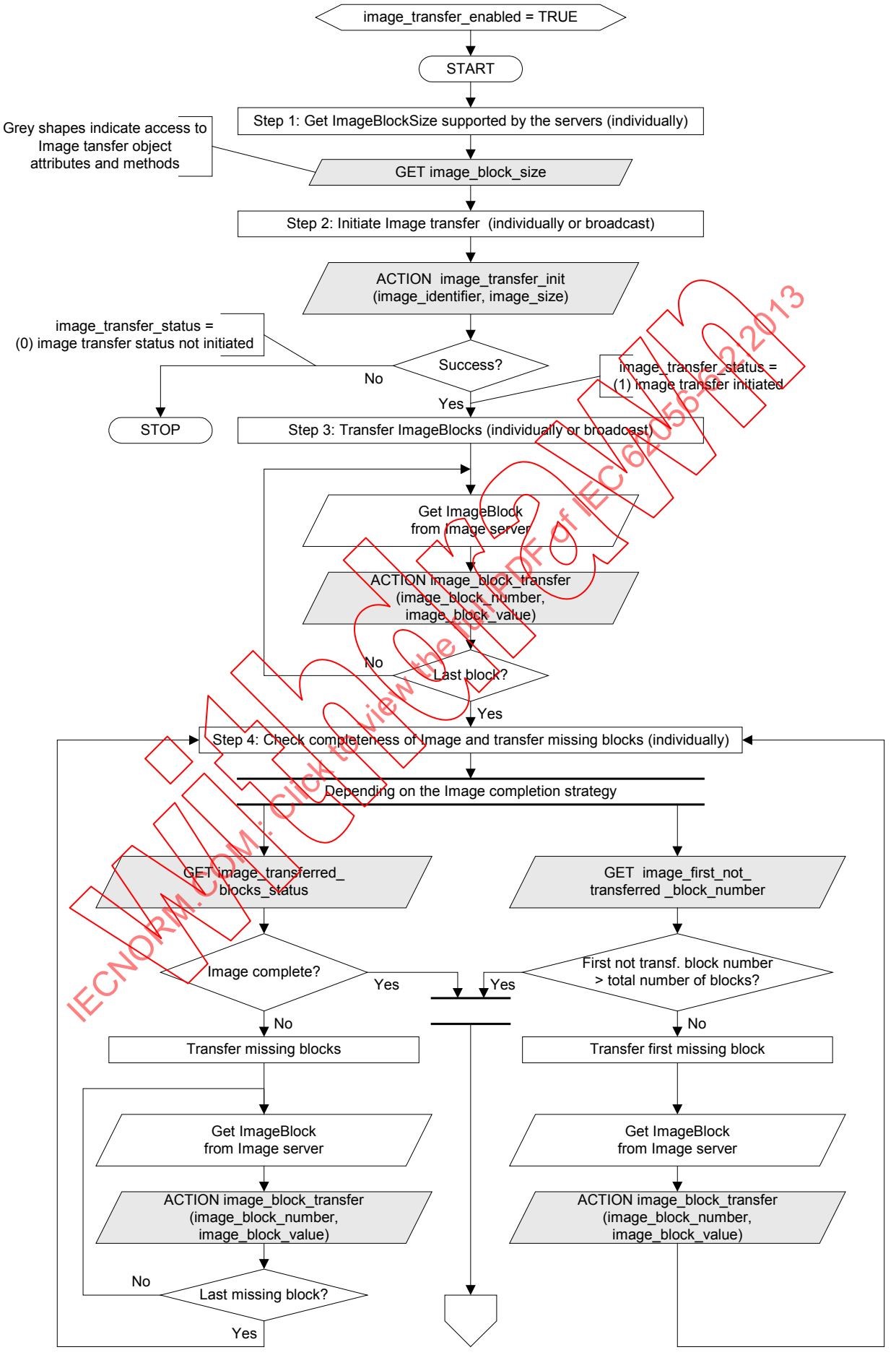
Step 7: Activate Image: In the seventh and final step, the Image is activated. An Image transferred may hold one or several Images to be activated.

The Image activation is done by invoking the *image_activate* method, either by the client or by the server itself. If the activation is done without a previous verification, then verification is done implicitly as part of the activation. The result can be:

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

NOTE 6 For xDLMS services Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:—, Clause 8.

The result of the Image activation may be checked by the client by retrieving the value of the *image_transfer_status* attribute.



IEC 1105/13

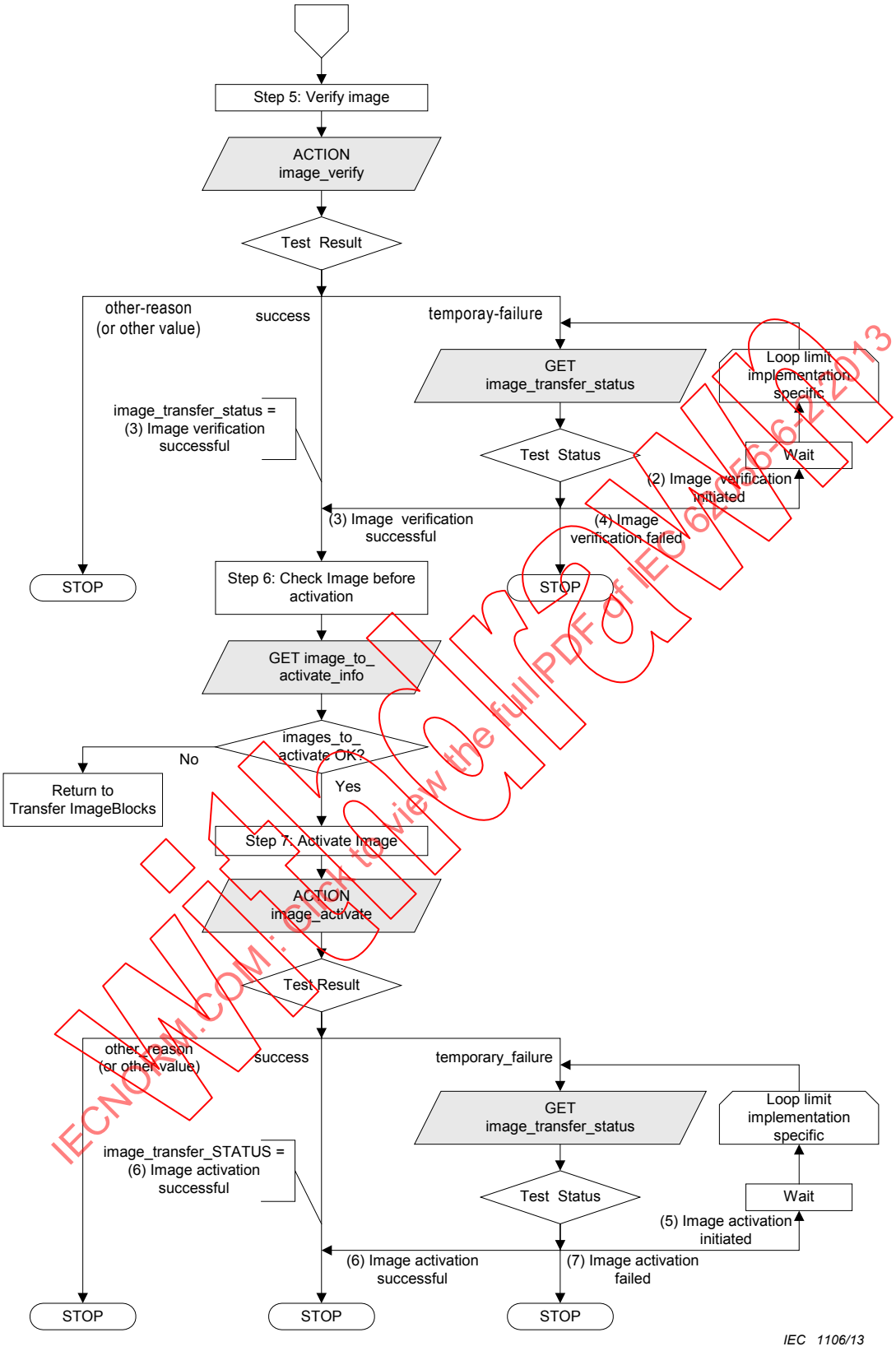


Figure 11 – Image transfer process flow chart

5.3.4.4 The Image transfer IC definition

Image transfer	**0…n**	**class_id = 18, version = 0**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**

1. logical_name	(static)	octet-string				x
2. image_block_size	(static)	double-long-unsigned				x + 0x08
3. image_transferred_blocks_status	(dyn.)	bit-string				x + 0x10
4. image_first_not_transferred_block_number	(dyn.)	double-long-unsigned				x + 0x18
5. image_transfer_enabled	(static)	boolean				x + 0x20
6. image_transfer_status	(dyn.)	enumerated				x + 0x28
7. image_to_activate_info	(dyn.)	array				x + 0x30
Specific methods		***m/o***				
1. image_transfer_initiate		m				x + 0x40
2. image_block_transfer		m				x + 0x48
3. image_verify		m				x + 0x50
4. image_activate		m				x + 0x58

Attribute description

logical_name

Identifies the “Image Transfer” object instance. See 6.2.26.

image_block_size

Holds the ImageBlockSize, expressed in octets, which can be handled by the server.

It should not exceed the ServerMaxReceivePduSize negotiated.

image_transferred_blocks_status

Provides information about the transfer status of each ImageBlock. Each bit in the bit-string provides information about one individual ImageBlock:

0 = Not transferred,
1 = Transferred

image_first_not_transferred_block_number

Provides the ImageBlockNumber of the first ImageBlock not transferred.

If the Image is complete, the value returned should be above the number of blocks calculated from the Image size and the ImageBlockSize.

image_transfer_enabled

Controls enabling the Image transfer process. The methods can be invoked successfully only if the value of this attribute is TRUE.

boolean: FALSE = Disabled,
TRUE = Enabled

image_transfer_status

Holds the status of the Image transfer process.

enum:
(0) Image transfer not initiated,
(1) Image transfer initiated,
(2) Image verification initiated,
(3) Image verification successful,
(4) Image verification failed,
(5) Image activation initiated,
(6) Image activation successful,

(7) Image activation failed

image_to_activate_info	Provides information on the Image(s) ready for activation. It is generated as the result of the Image verification. The client may check this information before activating the Image(s).
	
	array image_to_activate_info_element
	
	image_to_activate_info_element::= structure
	
	{
	image_size: double-long-unsigned,
	image_identification: octet-string,
	image_signature: octet-string
	}
	where
	• image_size is the size of the Image(s) to be activated, expressed in octets;
	• image_identification is the identification of the Image(s) to be activated, and may contain information like manufacturer, device type, version information, etc.;
	• image_signature is the signature of the Image(s) to be activated.
	
	NOTE 1 This attribute contains information on the Image(s) that will be activated. In the case of electricity, the information on the currently active Image(s) is held by the object 1.b.0.2.0.255, Configuration program version number (aka Active firmware version). The Active firmware signature is held by the object 1.b.0.2.8.255. For other media, see the relevant clauses.

Method description	

image_transfer_initiate (data)	Initializes the Image transfer process.
	data::= structure
	{
	image_identifier: octet-string,
	image_size: double-long-unsigned
	}
	where:
	image_identifier identifies the Image to be transferred;
	image_size holds the ImageSize, expressed in octets.

image_block_transfer (data)	Transfers one block of the Image to the server.
	data::= structure
	{
	image_block_number: double-long-unsigned,
	image_block_value: octet-string
	}
	NOTE 2 As specified in 5.3.4.2, the first ImageBlock sent is block 0.

image_verify (data)	Verifies the integrity of the Image before activation. data::= integer (0) The result of the invocation of this method may be success, temporary-failure or other-reason. If it is not success, then the result of the verification can be learned by retrieving the value of the image_transfer_status attribute. NOTE 3 For xDLMS services, Action-Result / Data-Access-Result codes are

	specified in IEC 62056-5-3:—, Clause 8.
image_activate (data)	Activates the Image(s). data::= integer (0) If the Image transferred has not been verified before, then this is done as part of the Image activation. The result of the invocation of this method may be success, temporary-failure or other-reason. If it is not success, then the result of the activation can be learned by retrieving the value of the image_transfer_status attribute. NOTE 4 For xDLMS services Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:—, Clause 8.

5.3.5 Security setup (class_id: 64, version: 0)

Instances of this IC contain the necessary information on the security policy applicable and the security suite in use within a particular AA, between two systems identified by their client system title and server system title respectively. They also contain methods to increase the level of security and to transfer the global keys. See also IEC 62056-5-3:—, Clause 5.

Security setup		**0…n**	**class_id = 64, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. security_policy	(static)	enum				x + 0x08
3. security_suite	(static)	enum				x + 0x10
4. client_system_title	(dyn.)	octet-string				x + 0x18
5. server_system_title	(static)	octet-string				x + 0x20
Specific methods		***m/o***				
1. security_activate		o				x + 0x28
2. global_key_transfer		o				x + 0x30

Attribute description

logical_name	Identifies the “Security setup” object instance. See 6.2.25.
security_policy	Enforces authentication and/or encryption algorithm provided with security_suite. enum: (0) nothing, (1) all messages to be authenticated, (2) all messages to be encrypted, (3) all messages to be authenticated and encrypted
security_suite	Specifies authentication, encryption and key transport algorithm. enum: (0) AES-GCM-128 for authenticated encryption and AES-128 for key wrapping

client_system_title	Carries the (current) client system title: • in the S-FSK PLC environment, the active initiator sends its system title using the CIASE protocol; NOTE 1 It is also held by the active_initiator attribute of the S-FSK Active initiator object; see 5.8.5; • during confirmed or unconfirmed AA establishment, it is carried by the calling-AP-title field of the AARQ APDU; If a client system title has already been sent during a registration process, like in the case of the S-FSK PLC profile, the client system title carried by the AARQ APDU should be the same. If not, the AA shall be rejected and appropriate diagnostic information shall be sent. • in a pre-established AA, it can be written by the client using an unsecured SET / Write service.
server_system_title	Carries the server system title. • in the S-FSK PLC environment, the server sends its system title during the discover process, using the CIASE protocol; • during confirmed AA establishment, it is carried by the responding-AP-title field of the AARE APDU. This attribute shall be read only.
Method description	
security_activate (data)	Activates and strengthens the security policy: enum: (0) nothing, (1) all messages to be authenticated, (2) all messages to be encrypted, (3) all messages to be authenticated and encrypted NOTE 2 The security policy can only be strengthened.

global_key_ transfer (data)	Updates one or more global keys.
	The *data* parameter includes wrapped key data. Key data include the key identifiers and the keys themselves.
	array key_data
	key_data::= structure
	{
	key_id: enum (0) global unicast encryption key,
	(1) global broadcast encryption key,
	(2) authentication key
	key_wrapped: octet-string
	}
	The key wrapping algorithm is as specified by the security suite. The KEK is the master key.

5.4 Interface classes for time- and event bound control

5.4.1 Clock (class_id: 8, version: 0)

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

The date information includes the elements year, month, day of month and day of week. The time information includes the elements hour, minutes, seconds, hundredths of seconds, and

the deviation of the local time from UTC. The daylight saving time function modifies the deviation of local time to UTC depending on the attributes. The start and end point of that function is normally set once. An internal algorithm calculates the real switch point depending on these settings. See Figure 12.

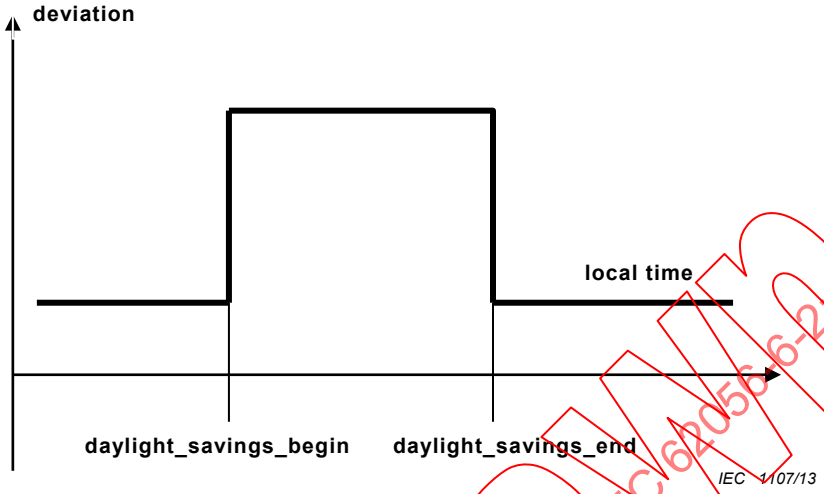


Figure 12 – The generalized time concept

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

Attribute description

logical_name	Identifies the “Clock” object instance. See 6.2.4.
time	Contains the meter’s local date and time, its deviation to UTC and the status. See 4.6.1. 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 4.6.1 for date_time
time_zone	The deviation of local, normal time to UTC in minutes.
status	The status is equal to the status read in time. See 4.6.1. unsigned, formatted as set in 4.6.1 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 4.6.1 for date_time
daylight_savings_ end	See above. octet-string, formatted as set in 4.6.1 for date_time
daylight_savings_ deviation	Contains the number of minutes by which the deviation in generalized time shall be corrected at daylight savings begin. integer: Deviation range of up to ± 120 min
daylight_savings_ enabled	boolean: TRUE = DST enabled FALSE = DST disabled
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 4.6.1 for *date_time*
shift_time (data)	Shifts the time by *n* (-900 <= *n* <= 900) s. data::= long

5.4.2 Script table (class_id: 9, version: 0)

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

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

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

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

Script table		**0...n**	**class_id = 9, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. scripts	(static)	array				x + 0x08
Specific methods		***m/o***				
1. execute(data)		m				x + 0x20

Attribute description

logical_name	Identifies the “Script table” object instance. See 6.2.6.

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

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

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

5.4.3 Schedule (class_id: 10, version: 0)

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

Table 5 – Schedule

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

Table 6 – Special days table

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

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

Attribute description

logical_name Identifies the "Schedule" object instance. See 6.2.8.

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
}
```

where:

- script_logical_name: defines the logical name of the “Script table” object;
- script_selector: defines the script_identifier of the script to be executed;
- switch_time accepts wildcards to define repetitive entries. The format of the octet-string follows the rules set in 4.6.1 for time;
- validity_window defines a period in minutes, in which an entry shall be processed after power fail. (time between defined switch_time and actual power_up) 0xFFFF: the script shall be processed any time;
- exec_weekdays defines the days of the week on which the entry is valid;
- exec_specdays perform the link to the IC “Special days table”, day_id;
- begin_date and end_date define the date period in which the entry is valid (wildcards are allowed). The format follows the rules set in 4.6.1 for date.

Method description

enable/disable (data)

Sets the disabled bit of range A entries to true and then enables the entries of range B.

data::= structure
{
    firstIndexA: long-unsigned,
    lastIndexA: long-unsigned,
    firstIndexB: long-unsigned,
    lastIndexB: long-unsigned
}

firstIndexA	first index of the range that is disabled
lastIndexA	last index of the range that is disabled
firstIndexB	first index of the range that is enabled
lastIndexB	last index of the range that is enabled
firstIndexA/B < lastIndexA/B:	all entries of the range A/B are disabled/enabled
firstIndexA/B == lastIndexA/B:	one entry is disabled/enabled
firstIndexA/B > lastIndexA/B:	nothing disabled/enabled
firstIndexA/B and lastIndexA/B > 9999:	no entry is disabled/enabled

insert (data)

Inserts a new entry in the table. If the index of the entry exists already, the existing entry is overwritten by the new entry.

entry: schedule_table_entry

data: corresponding to entry

delete (data)

Deletes a range of entries in the table.

data::= structure
{
    firstIndex: long-unsigned,
    lastIndex: long-unsigned
}

firstIndex	first index of the range that is deleted

lastIndex	last index of the range that is deleted
firstIndex < lastIndex:	all entries of the range A/B are deleted
firstIndex::= lastIndex:	one entry is deleted
firstIndex > lastIndex:	nothing deleted

Remarks concerning "inconsistencies" in the table entries:

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

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

- **Recovery after power failure**

After a power failure, the whole schedule is processed to execute all the necessary scripts that would get lost during a power failure. For this, the entries that were not executed during the power failure shall 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 (by writing to the attribute "time" of the "Clock" object);

b) time setting backward (by writing to the attribute "time" of the "Clock" object);

c) time synchronization (using the method "adjust_to_quarter" of the "Clock" object);

d) daylight saving action.

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

- **Time setting forward**

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

- **Time setting backward**

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

- **Time synchronization**

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

- **Daylight saving**

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

5.4.4 Special days table (class_id: 11, version: 0)

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

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

Attribute description

logical_name	Identifies the “Special days table” object instance. See 6.2.7.

entries	Specifies a special day identifier for a given date. The date may have wildcards for repeating special days like Christmas.
	array spec_day_entry
	spec_day_entry::= structure
	{
	index: long-unsigned,
	specialday_date: octet-string,
	day_id: unsigned
	}
	where:
	• specialday_date formatting follows the rules set in 4.6.1 for *date*;
	• the range of the day_id shall match the length of the bitstring exec_specdays in the related object of IC “Schedule”.

Method description

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

5.4.5 Activity calendar (class_id: 20, version: 0)

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

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

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

The “Activity calendar” object defines the activation of certain scripts, which can perform different activities inside the logical device. The interface to the IC “Script table” is the same as for the IC “Schedule” (see 5.4.3).

If an instance of the IC “Special days table” (see 5.4.4) is available, relevant entries there take precedence over the “Activity calendar” object driven selection of a day profile. The day profile referenced in the “Special days table” activates the day_schedule of the day_profile_table in the “Activity calendar” object by referencing through the day_id.

Activity calendar		**0...n**	**class_id = 20, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. calendar_name_active	(static)	octet-string				x + 0x08
3. season_profile_active	(static)	array				x + 0x10
4. week_profile_table_active	(static)	array				x + 0x18
5. day_profile_table_active	(static)	array				x + 0x20
6. calendar_name_passive	(static)	octet-string				x + 0x28
7. season_profile_passive	(static)	array				x + 0x30
8. week_profile_table_passive	(static)	array				x + 0x38
9. day_profile_table_passive	(static)	array				x + 0x40
10. activate_passive_calendar_time	(static)	octet-string				x + 0x48
Specific methods		***m/o***				
1. activate_passive_calendar (data)		o				x + 0x50

Attribute description

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

logical_name	Identifies the “Activity calendar” object instance. See 6.2.9.
calendar_name	Typically contains an identifier, which is descriptive to the set of scripts activated by the object.

season_profile	Contains a list of seasons defined by their starting date and a specific week_profile to be executed. The list is sorted according to season_start (in increasing order);
	array season
	season::= structure
	{
	season_profile_name: octet-string,
	season_start: octet-string,
	week_name: octet-string
	}

where:

- season_profile_name is a user defined name identifying the current season_profile;
- season_start defines the starting time of the season, formatted as set in 4.6.1 for *date_time*. In season_start, wildcards are allowed. If all fields are wildcards, the season will never start. When using wildcards, special care has to be taken to avoid conflicting parametrization, i.e. that the season_start of two different seasons is the same.

NOTE The current season is terminated by the season_start of the next season.

week_name defines the week_profile active in this season

week_profile_table

Contains an array of week_profiles to be used in the different seasons. For each week_profile, the day_profile for every day of a week is identified.

```
array  week_profile
week_profile::= structure
{
     week_profile_name:     octet-string,
     monday:                day_id,
     tuesday:               day_id,
     wednesday:        day_id,
     thursday:              day_id,
     friday:           day_id,
     saturday:              day_id,
     sunday:                day_id
}
day_id: unsigned
```

where:

- week_profile_name is a user defined name identifying the current week_profile;
- Monday defines the day_profile valid on Monday;
- ...
- Sunday defines the day_profile valid on Sunday.

day_profile_table

Contains an array of day_profiles, identified by their day_id. For each day_profile, a list of scheduled actions is defined by a script to be executed and the corresponding activation time (start_time). The list is sorted according to start_time.

```
array  day_profile
day_profile::= structure
{
  day_id:          unsigned,
  day_schedule:    array day_profile_action
}

day_profile_action::= structure
{
  start_time:                octet-string,
  script_logical_name:       octet-string,
  script_selector:      long-unsigned
}
```

where:

- day_id is a user defined identifier, identifying the current day_profile;
- start_time: defines the time when the script is to be executed (no wildcards); the format follows the rules set in 4.6.1 for *time*;
- script_logical_name: defines the logical name of the “Script table” object;
- script_selector: defines the script_identifier of the script to be executed.

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 is not allowed. octet-string, formatted as set in 4.6.1 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)

5.4.6 Register monitor (class_id: 21, version: 0)

This IC allows modelling the function of monitoring of values modelled by “Data”, “Register”, “Extended register” or “Demand register” objects. It allows specifying thresholds, the value monitored, and a set of scripts (see 5.4.2) that are executed when the value monitored crosses a threshold.

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

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

Attribute description

logical_name	Identifies the “Register monitor” object instance. See 6.2.12 and 6.3.10.
thresholds	Provides the threshold values to which the attribute of the referenced register is compared. array threshold threshold: The threshold is of the same type as the monitored attribute of the referenced object.
monitored_value	Defines which attribute of an object is to be monitored. Only values with simple data types are allowed.

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

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 value of the monitored register crosses the threshold in the upwards direction;
- action_down defines the action when the attribute value of the monitored register crosses the threshold in the downwards direction.

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

5.4.7 Single action schedule (class_id: 22, version: 0)

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

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

Attribute description

logical_name

Identifies the “Single action schedule” object instance. See 6.2.11.

executed_script

Contains the logical name of the “Script table” object and the script selector of the script to be executed.

script::= structure
{
script_logical_name: octet-string,
script_selector: long-unsigned
}
Script_logical_name and script_selector define the script to be executed.

type	enum:	(1) size of execution_time = 1; wildcard in *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 and the date when the script is executed.
	array execution_time_date
	execution_time_date::= structure
	{
	time: octet-string,
	date: octet-string
	}
	The two octet-strings contain *time* and *date*, in this order; *time* and *date* are formatted as defined in 4.6.1.
	WILDCARDS in *time* are not allowed; seconds and hundredths of seconds shall be zero.	

5.4.8 Disconnect control (class_id: 70, version: 0)

Instances of the Disconnect control IC manage an internal or external disconnect unit of the meter (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely – the premises of the consumer to / from the supply. The state diagram and the possible state transitions are shown in Figure 13.

Disconnect and reconnect can be requested:

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

The states and state transitions of the Disconnect control IC are shown in Table 7. The possible state transitions depend on the control mode. The Disconnect control object doesn't feature a memory, i.e. any commands are executed immediately.

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

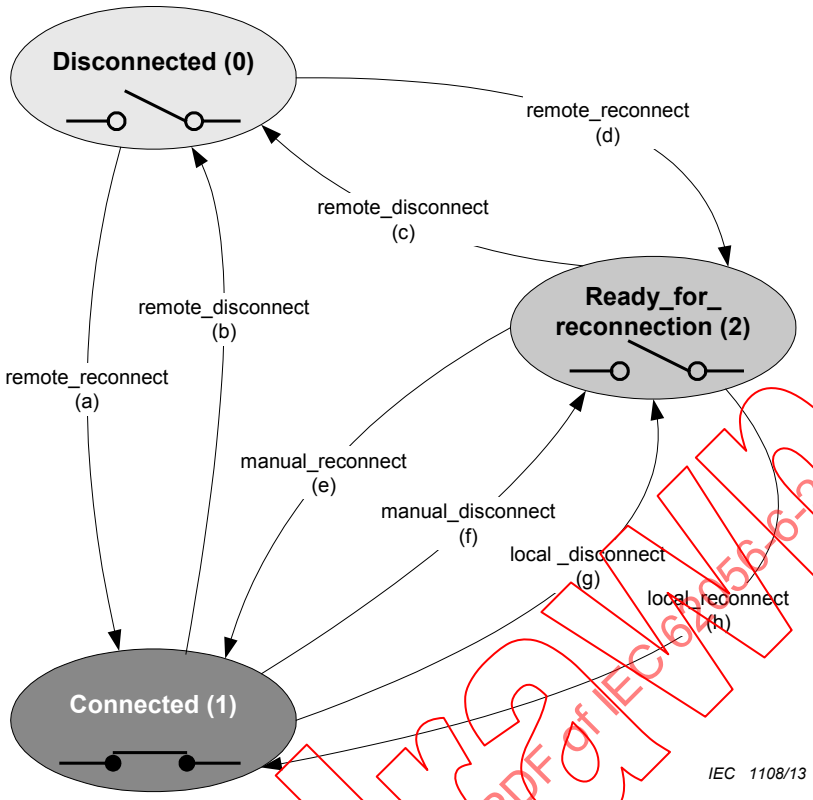


Figure 13 – State diagram of the Disconnect control IC

Table 7 – Disconnect control IC – states and state transitions

States		
State number	**State name**	**State description**
0	Disconnected	The output_state is set to FALSE and the consumer is disconnected.
1	Connected	The output_state is set to TRUE and the consumer is connected.
2	Ready_for_reconnection	The output_state is set to FALSE and the consumer is disconnected.
State transitions		
Transition	**Transition name**	**State description**
a	remote_reconnect	Moves the Disconnect control object from the Disconnected (0) state directly to the Connected (1) state without manual intervention.
b	remote_disconnect	Moves the Disconnect control object from the Connected (1) state to the Disconnected (0) state.
c	remote_disconnect	Moves the Disconnect control object from the Ready_for_reconnection (2) state to the Disconnected (0) state.
d	remote_reconnect	Moves the Disconnect control object from the Disconnected (0) state to the Ready_for_reconnection (2) state. From this state, it is possible to move to the Connected (2) state via the manual_reconnect transition (e) or local_reconnect transition (h).
e	manual_reconnect	Moves the Disconnect control object from the Ready_for_connection (2) state to the Connected (1) state.
f	manual_disconnect	Moves the Disconnect control object from the Connected (1) state to the Ready_for_connection (2) state. From this state, it is possible to move back to the Connected (2) state via the manual_reconnect transition (e) or local_reconnect transition (h).
g	local_disconnect	Moves the Disconnect control object from the Connected (1) state to the Ready_for_connection (2) state. From this state, it is possible to move back to the Connected (2) state via the manual_reconnect transition (e) or local_reconnect transition (h). NOTE 1 Transitions f) and g) are essentially the same, but their trigger is different.
h	local_reconnect	Moves the Disconnect control object from the Ready_for_connection (2) state to the Connected (1) state NOTE 2 Transitions e) and h) are essentially the same, but their trigger is different.

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

Attribute description

logical_name	Identifies the “Disconnect control” object instance. See 6.2.18 and 6.2.32.
output_state	Shows the actual physical state of the disconnect unit, i.e. if an electricity breaker or a gas valve is open or closed. boolean: TRUE = (BOOLEAN 1) = Connected; FALSE = (BOOLEAN 0) = Disconnected. NOTE 1 In electricity metering, the supply is connected when the disconnector device is closed. NOTE 2 In gas and water metering, the supply is connected when the valve is open.
control_state	Shows the internal state of the disconnect control object. enum: (0) Disconnected, (1) Connected, (2) Ready_for_reconnection
control_mode	Configures the behaviour of the disconnect control object for all triggers, i.e. the possible state transitions. enum: (0) None. The disconnect control object is always in 'connected' state, (1) Disconnection: Remote (b, c), manual (f), local (g) Reconnection: Remote (d), manual (e), (2) Disconnection: Remote (b, c), manual (f), local (g) Reconnection: Remote (a), manual (e), (3) Disconnection: Remote (b, c), manual (-), local (g) Reconnection: Remote (d), manual (e), (4) Disconnection: Remote (b, c), manual (-), local (g) Reconnection: Remote (a), manual (e) (5) Disconnection: Remote (b, c), manual (f), local (g) Reconnection: Remote (d), manual (e), local (h), (6) Disconnection: Remote (b, c), manual (-), local (g) Reconnection: Remote (d), manual (e), local (h) NOTE 3 Local disconnection is always possible unless the corresponding trigger is inhibited.

Method description

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

5.4.9 Limiter (class_id: 71, version: 0)

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

The threshold value can be normal or emergency threshold. The *emergency threshold* is activated via the *emergency profile* defined by *emergency profile id*, *emergency activation*

time, and *emergency duration*. The *emergency profile id* element is matched to an *emergency profile group id*: this mechanism enables the activation of the emergency threshold only for a specific emergency group.

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

Attribute description

logical_name

Identifies the "Limiter" object instance. See 6.2.13.

monitored_value

Defines an attribute of an object to be monitored. Only attributes with simple data types are allowed.

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

threshold_active

Provides the active threshold value to which the attribute monitored is compared.

threshold: The threshold is of the same type as the attribute monitored.

threshold_normal

Provides the threshold value to which the attribute monitored is compared when in normal operation.

threshold: see above

threshold_emergency

Provides the threshold value to which the attribute monitored is compared when an emergency profile is active.

threshold: see above

min_over_threshold_duration

Defines minimal over threshold duration in seconds required to execute the over threshold action.

min_under_threshold_duration

Defines minimal under threshold duration in seconds required to execute the under threshold action.

emergency_profile

An emergency_profile is defined by three elements: emergency_profile_id, emergency_activation_time, emergency_duration.

An emergency profile is activated if the emergency_profile_id

	element matches one of the elements on the emergency_profile _group_id_list, and time matches the emergency_activation_time and emergency_duration element:
	
	emergency_profile::= structure
	{
	emergency_profile_id: long-unsigned,
	emergency_activation_time: octet-string,
	emergency_duration: double-long-unsigned
	}
	emergency_activation_time defines the date and time when the emergency_profile activated. The octet-string is encoded as specified in 4.6.1 for date_time.
	
	emergency_duration defines the duration in seconds, for which the emergency_profile is activated.
	
	When an emergency profile is active, the emergency_profile_active attribute is set to TRUE.

emergency_profile_group_ id_list	Defines a list of group id-s of the emergency profile. The emergency profile can be activated only if emergency_profile_id element of the emergency_profile matches one of the elements on the emergency_profile_group_id-list: array emergency_profile_group_id emergency_profile_group_id: long-unsigned
emergency_profile_active	Indicates that the emergency_profile is active.

actions	Defines the scripts to be executed when the monitored value crosses the threshold for minimal duration time.
	action::= structure
	{
	action_over_threshold: action_item,
	action_under_threshold: action_item
	}
	where:
	• action_over_threshold defines the action when the value of the attribute monitored crosses the threshold in upwards direction and remains over threshold for minimal over threshold duration time;
	• action_under_threshold defines the action when the value of the attribute monitored crosses the threshold in the downwards direction and remains under threshold for minimal under threshold duration time.
	
	action_item::= structure
	{
	script_logical_name: octet-string,
	script_selector: long-unsigned
	}

5.4.10 Sensor manager interface class (class_id:67, version: 0)

5.4.10.1 General

Most measuring instruments under the scope of the MID operate with dedicated sensors (transducers and transmitters) connected to the processing unit. These sensors have to be

permanently supervised concerning their functioning and limits to fulfil the metrological requirements for subsequent calculation of monetary values.

In addition, the measured values have to be monitored. These values may be related to a physical quantity – raw values of voltage, current, resistance, frequency, digital output – provided by the sensor, and the measured quantities resulting from the processing of the information provided by the sensor.

It is necessary to monitor and often to log the relevant values in order to obtain diagnostic information that allows:

- the identification of the sensor device;
- the connection and the sealing status of the sensor;
- the configuration of the sensors;
- the monitoring of the operation of the sensors;
- the monitoring of the result of the processing.

The Sensor manager interface class allows managing detailed information related to a sensor by a single object.

For simpler sensors / devices, already existing COSEM objects – identifying the sensors, holding measurement values and monitoring those measurement values – can be used.

5.4.10.2 The Sensor manager interface class specification

Instances of the “Sensor manager” IC manage complex information related to sensors. They also allow monitoring the raw data and the processed value, derived by processing the raw-data using appropriate algorithms as required by the particular application. This IC includes a number of functions:

- nameplate data of the sensor and site information (attributes 2 to 6);
- an “Extended register” function for the raw data (attributes 7 to 10);
- a “Register monitor” function for the raw data (attributes 11-12);

NOTE 1 Not every raw data (e.g. the voltage output of a pressure sensor) has its own OBIS code / object. This is the reason to include raw data in the Sensor manager class.

- a “Register monitor” function for the processed value (attributes 13 to 15).

NOTE 2 Not all “modules” are necessarily present. The attributes not used are possibly not implemented or not accessible.

Sensor manager		**0...n**	**class_id = 67, version = 0**			
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. serial_number	(dyn,)	octet-string				x + 0x08
3. metrological_identification	(dyn.)	octet-string				x + 0x10
4. output_type	(dyn.)	enum				x + 0x18
5. adjustment_method	(dyn.)	octet-string				x + 0x20
6. sealing_method	(dyn.)	enum				x + 0x28
7. raw_value	(dyn.)	CHOICE				x + 0x30
8. scaler_unit	(dyn.)	structure				x + 0x38

Sensor manager		0...n	class_id = 67, version = 0			
9. status	(dyn.)	CHOICE				x + 0x40
10. capture_time	(dyn.)	date_time				x + 0x48
11. raw_value_thresholds	(dyn.)	array				x + 0x50
12. raw_value_actions	(dyn.)	array				x + 0x58
13. processed_value	(dyn.)	processed_value_ definition				x + 0x60
14. processed_value_thresholds	(dyn.)	array				x + 0x68
15. processed_value_actions	(dyn.)	array				x + 0x70
Specific methods		***m/o***				
1. reset (data)		*o*				x + 0x80

Attribute description

logical_name

Identifies the “Sensor manager” object instance.

serial_number

Identifies the sensor (->site information).

NOTE 3 For a simple sensor, the serial number can be held by Data objects with appropriate OBIS codes if this is the only information required. See Gas related general purpose objects, Configuration objects.

metrological_identification

Describes metrologically relevant information of the sensor, e.g. metrological identifier, calibration date.

output_type

describes the physical output of the sensor (->site information)

enum
0 = not specified,
1 = 4–20 mA,
2 = 0–20 mA,
3 = 0–5 V,
4 = 0–10 V,
5 = Pt100,
6 = Pt500,
7 = Pt1 000,
8 = HART
128 = manufacturer specific

All other values are reserved.

adjustment_method

Describes the sensor adjustment method, e.g. by 3-Measuring-point-equation.

sealing_method

Type of seals applied to the sensor.

enum
0 = none,
1 = mechanical (e.g. wire or protective sticker);
2 = electronic (e.g. contact);
3 = software (e.g. password protection)

raw_value

Physical value from the sensor (e.g. voltage from a pressure to voltage converter).

For the possible data type choices, see the “Register” IC.

scaler_unit

The scaler and the unit of the raw_value.

For the definition of scaler_unit, see the “Register” IC.

status	Status of last raw value captured (for the definition of status, see the “Extended register” IC. The status keeps information like: sensor failure, sensor activated / inactivated. The meaning of the elements of the status shall be provided for each “Sensor manager” object instance.
capture_time	Provides a “Sensor manager” object specific date and time information showing when the value of the attribute "raw_value" has been captured. For the possible data type choices, see the “Extended register” interface class.
raw_value_thresholds	Provides the threshold values to which the raw_value attribute is compared. array threshold threshold: The threshold is of the same type as the raw-data.
raw_value_actions	Defines the scripts to be executed when the “raw_value” crosses the corresponding threshold. The attribute “raw_value_actions” has exactly the same number of elements as the attribute “raw_value_thresholds”. The ordering of the action_items corresponds to the ordering of the thresholds. For the specification of actions, see the “Register monitor” IC.

processed_value	References the attribute holding the processed value of the raw-data provided by the sensor.
	processed_value_definition::= structure
	{
	class_id: long_unsigned,
	logical_name: octet-string,
	attribute_index: integer
	}
	Note that more than one Sensor manager objects and processed value objects may belong to the same sensor.

processed_value_thresholds	Provides the threshold values to which the processed value is compared. array threshold threshold: The threshold is of the same type as the processed value.
processed_ value_actions	Defines the scripts to be executed when the processed value crosses the corresponding threshold. The attribute “processed_value_actions” has exactly the same number of elements as the attribute “processed_value_thresholds”. The ordering of the action_items corresponds to the ordering of the thresholds. For the specification of actions, see the “Register monitor” IC.

Method description

reset (data)	This method resets the raw_value to the default value. The default value is an instance specific constant. The attribute capture_time is set to the time of the reset execution. data::= integer(0)

5.4.10.3 Example for absolute pressure sensor

Figure 14 illustrates the definition of relevant upper and lower thresholds.

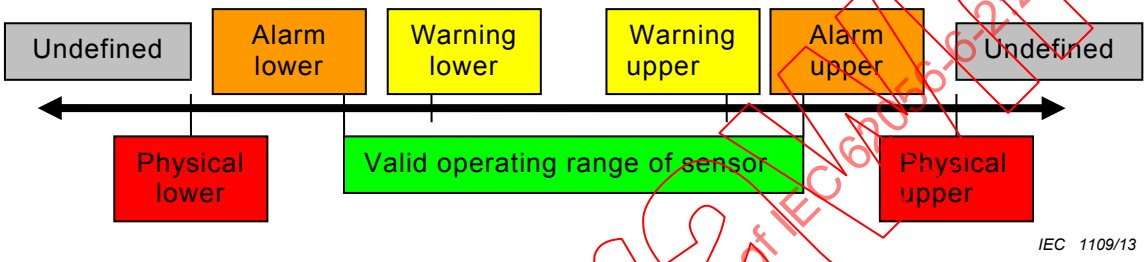


Figure 14 – Definition of upper and lower thresholds

Table 8 shows an example of the various thresholds.

Table 8 – Explicit presentation of threshold value arrays

Threshold	Physical lower	Physical upper	Alarm lower	Alarm upper	Warning lower	Warning upper
value	1,0	5,5	1,2	5,0	1,4	4,8
scaler_unit	1, Volt	1, Volt	1, bar	1, bar	1, bar	1, bar

Table 9 shows an example of actions to be performed when the thresholds are crossed.

Table 9 – Explicit presentation of action_sets

action_set	Physical lower	Physical upper	Alarm lower	Alarm upper	Warning lower	Warning upper
action_up	clr_phys_alarm_bit	set_phys_alarm_bit	clr_alarm_bit	set_alarm_bit	clr_warn_bit	set_warn_bit
action_down	set_phys_alarm_bit	clr_phys_alarm_bit	set_alarm_bit	clr_alarm_bit	set_warn_bit	clr_warn_bit

5.5 Interface classes for setting up data exchange via local ports and modems

5.5.1 IEC local port setup (class_id: 19, version: 1)

This IC allows modelling the configuration of communication ports using the protocols specified in IEC 62056-21. Several ports can be configured.

IEC local port setup		0...n	class_id = 19, version = 1			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. default_mode	(static)	enum				x + 0x08
3. default_baud	(static)	enum				x + 0x10
4. prop_baud	(static)	enum				x + 0x18
5. response_time	(static)	enum				x + 0x20
6. device_addr	(static)	octet-string				x + 0x28
7. pass_p1	(static)	octet-string				x + 0x30
8. pass_p2	(static)	octet-string				x + 0x38
9. pass_w5	(static)	octet-string				x + 0x40
Specific methods		***m/o***				

Attribute description

logical_name Identifies the "IEC local port setup" object instance. See 6.2.14.

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 IC are not applicable,
(2) protocol not specified. Using this enumeration value, attribute 4), prop_baud, is used for setting the communication speed on the port. All other attributes are not applicable.

default_baud Defines the baud rate for the opening sequence.

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

prop_baud Defines the baud rate to be proposed by the meter.

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

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

5.5.2 IEC HDLC setup (class_id: 23, version: 1)

This IC allows modelling and configuring communication channels according to IEC 62056-46. Several communication channels can be configured.

IEC HDLC setup		**0...n**	**class_id = 23, version = 1**			
Attributes		***Data type***	**Min.**	**Max.**	**Def.**	**Short name**
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. window_size_transmit	(static)	unsigned	1	7	1	x + 0x10
4. window_size_receive	(static)	unsigned	1	7	1	x + 0x18
5. max_info_field_length_transmit	(static)	long-unsigned	32	2030	128	x + 0x20
6. max_info_field_length_receive	(static)	long-unsigned	32	2030	128	x + 0x28
7. inter_octet_time_out	(static)	long-unsigned	20	6000	25	x + 0x30
8. inactivity_time_out	(static)	long-unsigned	0		120	x + 0x38
9. device_address	(static)	long-unsigned	0x0010	0x3FFD		x + 0x40
Specific methods		***m/o***				

NOTE 1 The maximum value of the attributes max_info_field_length_transmit and max_info_field_length_receive has been increased from 128 to 2 030 for efficiency reasons.

NOTE 2 A max_info_field_length_ receive of 128 bytes is needed to ensure a minimal performance.

NOTE 3 The maximum value of the inter-octet-time-out attribute has been increased from 1 000 ms to 6 000 ms in order to allow using communication media, where long delays can occur. The default value has been changed to 25 ms to align with 6.4.4.3.3 of IEC 62056-46:2002.

Attribute description

Logical_name	Identifies the “IEC HDLC setup” object instance. See 6.2.16.
comm_speed	The communication speed supported by the corresponding port.

	enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol.
window_size_ transmit	The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated.
window_size_ receive	The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated.
max_info_length_ transmit	The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.
max_info_length_ receive	The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.
inter_octet_time_ou t	Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame.
inactivity_time_out	Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection. When this value is set to 0, this means that the inactivity_time_out is not operational.
device_address	Contains the physical device address of a device. In the case of one byte addressing: 0x00 NO_STATION Address, 0x01…0x0F Reserved for future use, 0x10…0x7D Usable address space, 0x7E ‘CALLING’ device address, 0x7F Broadcast address In the case of two byte addressing: 0x0000 NO_STATION address, 0x0001..0x000F Reserved for future use, 0x0010..0x3FFD Usable address space, 0x3FFE ‘CALLING’ physical device address, 0x3FFF Broadcast address

5.5.3 IEC twisted pair (1) setup (class_id: 24, version: 0)

This IC allows modelling and configuring communication channels according to Draft IEC 62056-3-1[6]. Several communication channels can be configured.

IEC twisted pair (1) setup		0...n	class_id = 24, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. secondary_address	(static)	octet-string				x + 0x08
3. primary_address_list	(static)	primary_address_list_type				x + 0x10
4. tabi_list	(static)	tabi_list_type				x + 0x18
5. fatal_error	(dyn.)	enum				x + 0x20
Specific methods		***m/o***				

Attribute description

logical_name

Identifies the “IEC twisted pair setup” object instance. See 6.2.17.

secondary_address

Memorizes the ADS of the secondary station (see Draft IEC 62056-3-1) that corresponds to the real equipment.

octet-string (SIZE(6))

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 Draft IEC 62056-3-1).

```
primary_address_list_type::= array
{
        primary_address_element
}
primary_address_element::= octet-string (SIZE(1))
```

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 Draft IEC 62056-3-1).

```
tabi_list_type::= array tabi_element
tabi_element::= integer
```

fatal_error

Represents the last occurrence of one of the fatal errors of the protocols described in Draft IEC 62056-3-1.

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,
```

[6] To be published.

(7) e-EP-3F,
(8) e-EP-4F,
(9) e-EP-5F,
(10) e-EL-2F

5.5.4 Modem configuration (class_id: 27, version: 1)

This IC allows modelling the configuration and initialisation of modems used for data transfer from/to a device. Several modems can be configured.

Modem configuration		0...n	class_id = 27, version = 1			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. comm._speed	(static)	enum	0	9	5	x + 0x08
3. initialization_string	(static)	array				x + 0x10
4. modem_profile	(static)	array				x + 0x18
Specific methods		***m/o***				

Attribute description

logical_name

Identifies the “Modem configuration” object instance. See 6.2.5.

comm_speed

The communication speed between the device and the modem, not necessarily the communication speed on the WAN.

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

initialization_string

Contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features.

array initialization_string_element

initialization_string_element::= structure

{
request: octet-string,
response: octet-string,
delay_after_response: long-unsigned
}

If the array contains more than one initialization_string_element, the requests are sent in a sequence. The next request is sent after the expected response matching the previous request and waiting a delay_after_response time [ms], to allow the modem to execute the request.

NOTE It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed, the initialization string is empty.

modem_profile

Defines the mapping from Hayes standard commands/responses to modem specific strings.

array modem_profile_element

modem_profile_element::= octet-string

The modem_profile array shall contain the corresponding strings for the modem used in following order:

Element 0: OK,
Element 1: CONNECT,
Element 2: RING,
Element 3 NO CARRIER,
Element 4: ERROR,
Element 5: CONNECT 1 200,
Element 6 NO DIAL TONE,
Element 7: BUSY,
Element 8: NO ANSWER,
Element 9: CONNECT 600,
Element 10: CONNECT 2 400,
Element 11: CONNECT 4 800,
Element 12 CONNECT 9 600,
Element 13: CONNECT 14 400,
Element 14: CONNECT 28 800,
Element 15: CONNECT 33 600,
Element 16: CONNECT 56 000

5.5.5 Auto answer (class_id: 28, version: 0)

This IC allows modelling how the device manages the "Auto answer" function of the modem, i.e. answering of incoming calls. Several modems can be configured.

Auto answer		0...n	class_id = 28, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. listening_window	(static)	array				x + 0x10
4. status	(dyn.)	enum				x + 0x18
5. number_of_calls	(static)	unsigned				x + 0x20
6. number_of_rings	(static)	nr_rings_type				x + 0x28
Specific methods		***m/o***				

Attribute description

logical_name

Identifies the "Auto answer" object instance. See 6.2.5.

mode

Defines the working mode of the line when the device is auto answer.

	enum: (0) line dedicated to the device, (1) shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call, (2) shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date, (3) currently no modem connected, (200…255) manufacturer specific modes.

listening_window	Contains the start and end instant when the window becomes active (for the start instant), and inactive (for the end instant). The start_date defines implicitly the period.
	
	EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly …window management can be defined.
	
	array window_element
	
	window_element::= structure
	
	{
	start _time: octet-string,
	end_time: octet-string
	}
	start_time and end_time are formatted as set in 4.6.1 for *date_time*

status	Here is defined the status of the window. enum: (0) Inactive: the device will not manage any new incoming call. This status is automatically reset to Active when the next listening window starts, (1) Active: the device can answer to the next incoming call, (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.
number_of_calls	This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit.

number_of_rings Defines the number of rings before the meter connects the modem. Two cases are distinguished: number of rings within the window defined by attribute “listening_window” and number of rings outside the “listening_window”.

nr_rings_type::= structure
{
nr_rings_in_window: unsigned, (0: no connect in window)
nr_rings_out_of_window: unsigned (0: no connect out of window)
}

5.5.6 Auto connect (class_id: 29, version: 1)

This IC allows modelling the management of data transfer from the device to one or several destinations.

The messages to be sent, the conditions on which they shall be sent and the relation between the various modes, the calling windows and destinations are not defined here.

Depending on the mode, one or more instances of this IC may be necessary to perform the function of sending out messages.

Auto connect		**0...n**	**class_id = 29, version = 1**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. repetitions	(static)	unsigned				x + 0x10
4. repetition_delay	(static)	long-unsigned				x + 0x18
5. calling_window	(static)	array				x + 0x20
6. destination_list	(static)	array				x + 0x28
Specific methods		***m/o***				

Attribute description

logical_name Identifies the “Auto connect” object instance. See 6.2.5.

mode Defines the mode controlling the auto dial functionality concerning the timing, the message type to be sent and the infrastructure to be used.

enum:
(0) no auto dialling,
(1) auto dialling allowed anytime,
(2) auto dialling allowed within the validity time of the calling window,
(3) “regular” auto dialling allowed within the validity time of the calling window; “alarm” initiated auto dialling allowed anytime,
(4) SMS sending via Public Land Mobile Network (PLMN),
(5) SMS sending via PSTN,
(6) email sending,
(200..255) manufacturer specific modes

repetitions The maximum number of trials in the case of unsuccessful dialling attempts.

repetition_delay	The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated. repetition_delay 0 means delay is not specified.

calling_window	Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date defines implicitly the period.
	
	EXAMPLE When day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly …window management can be defined.
	
	array window_element
	window_element::= structure
	{
	start_time: octet-string,
	end_time: octet-string
	}
	start_time and end_time are formatted as set in 4.6.1 for *date_time*

destination_list	Contains the list of destinations (for example phone numbers, email addresses or their combinations) where the message(s) have to be sent under certain conditions. The conditions and their link to the elements of the array are not defined here. array destination destination::= octet-string

5.6 Interface classes for setting up data exchange via M-Bus

5.6.1 M-Bus slave port setup (class_id: 25, version: 0)

NOTE 1 The name of this IC has been changed from "M-BUS port setup" to "M-Bus slave port setup", to indicate that it serves to set up data exchange when a COSEM server communicates with a COSEM client using wired M-Bus.

This IC allows modelling and configuring communication channels according to EN 13757-2. Several communication channels can be configured.

M-Bus slave port setup	**0…n**	**class_id = 25, version = 0**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. default_baud (static)	enum	0	5	0	x + 0x08
3. avail_baud (static)	enum	0	7		x + 0x10
4. addr_state (static)	enum				x + 0x18
5. bus_address (static)	unsigned				x + 0x20
Specific methods	***m/o***				

Attribute description

logical_name	Identifies the "M-Bus slave port setup" object instance. See 6.2.18.
default_baud	Defines the baud rate for the opening sequence.

	enum: (0) 300 baud, (3) 2 400 baud, (5) 9 600 baud
avail_baud	Defines the baud rates that can be negotiated after startup. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud
addr_state	Defines whether or not the device has been assigned an address since last power up of the device. enum: (0) Not assigned an address yet, (1) Assigned an address either by manual setting, or by automated method.
bus_address	The currently assigned address on the bus for the device. NOTE 2 If no bus address is assigned, the value is 0.

5.6.2 M-Bus client (class_id: 72, version: 0)

Instances of this IC allow setting up M-Bus slave devices and exchanging data with them. Each M-Bus client object controls one M-Bus slave device. For details on the M-Bus dedicated application layer, see EN 13757-3.

The M-Bus client device may have one or more physical M-Bus interfaces, which can be configured using instances of the M-Bus master port setup IC, see 5.6.4.

An M-Bus slave device is identified with its Primary Address, Identification Number, Manufacturer ID etc. as defined in EN 13757-3:2004, Clause 5, *Variable Data respond*. These parameters are carried by the respective attributes of the M-Bus client IC, see 5.6.2.

Values to be captured from an M-Bus slave device are identified by the capture_definition attribute, containing a list of data identifiers (DIB, VIB) for the M-Bus slave device. The data are captured periodically or on an appropriate trigger. Each data element is stored in an M-Bus value object, or IC "Extended register". M-Bus value objects may be captured in M-Bus Profile generic objects, eventually along with other, non M-Bus specific objects.

Using the methods of M-Bus client objects, M-Bus slave devices can be installed and de-installed.

It is also possible to send data to M-Bus slave devices and to perform operations like resetting alarms, synchronizing the clock, transferring an encryption key, etc.

M-Bus client		**0...n**	**class_id = 72, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	***Short name***
1. logical_name	(static)	octet-string				x
2. mbus_port_reference	(static)	octet-string				x + 0x08
3. capture_definition	(static)	array				x + 0x10
4. capture_period	(static)	double-long-unsigned				x + 0x18
5. primary_address	(dyn.)	unsigned				x + 0x20

M-Bus client		0...n	class_id = 72, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	***Short name***
6. identification_number	(dyn.)	double-long-unsigned				x + 0x28
7. manufacturer_id	(dyn.)	long-unsigned				x + 0x30
8. version	(dyn.)	unsigned				x + 0x38
9. device_type	(dyn.)	unsigned				x + 0x40
10. access_number	(dyn.)	unsigned				x + 0x48
11. status	(dyn.)	unsigned				x + 0x50
12. alarm	(dyn.)	unsigned				x + 0x58
Specific methods		***m/o***				
1. slave_install (data)		o				x + 0x60
2. slave_deinstall (data)		o				x + 0x68
3. capture (data)		o				x + 0x70
4. reset_alarm (data)		o				x + 0x78
5. synchronize_clock (data)		o				x + 0x80
6. data_send (data)		o				x + 0x88
7. set_encryption_key (data)		o				x + 0x90
8. transfer_key (data)		o				x + 0x98

Attribute description

logical_name

Identifies the "M-Bus client" object instance. For logical names, see 6.2.18.

mbus_port_reference

Provides reference to an "M-Bus master port setup" object, used to configure an M-Bus port, each interface allowing to exchange data with one or more M-Bus slave devices.

capture_definition

Provides the capture_definition for M-Bus slave devices.

array capture_definition_element

capture_definition_element::= structure

```
{
    data_information_block:     octet-string,
    value_information_block:    octet-string
}
```

NOTE 1 The elements data_information_block and value_information_block correspond to Data Information Block (DIB) and Value Information Block (VIB) described in EN 13757-3:2004, 6.2 and Clause 7 respectively.

capture_period

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

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

primary_address

Carries the primary address of the M-Bus slave device, in the range 0...250.

If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the primary_address attribute. From this moment, the data exchange with

	the M-Bus slave device is possible. Otherwise, the slave_install method shall be used; see below. NOTE 2 The primary_address attribute cannot be used to store a desired primary address for an unconfigured slave device. If the primary address attribute is set, this means that the M-Bus client can immediately operate with this primary address, which is not the case with an unconfigured slave device.
identification_number	Carries the Identification Number element of the data header as specified in EN 13757-3:2004, 5.4. It is either a fixed fabrication number or a number changeable by the customer, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00 000 000 to 99 999 999. It can be preset at fabrication time with a unique number, but could be changeable afterwards, especially if, in addition, a unique and not changeable fabrication number (DIF = 0Ch, VIF = 78h) is provided, see 7.2.
manufacturer_id	Carries the Manufacturer Identification element of the data header as specified in EN 13757-3:2004, 5.5. It is coded unsigned binary with 2 bytes. This manufacturer_id is calculated from the ASCII code of the IEC 62056-21 manufacturer ID (three uppercase letters), using the formula specified in EN 13757-3:2004, 5.5.
version	Carries the Version element of the data header as specified in EN 13757-3:2004, 5.6. It specifies the generation or version of the meter and depends on the manufacturer. It can be used to make sure, that within each version number the identification # is unique.
device_type	Carries the Device type identification element of the data header as specified in EN 13757-3:2004, 5.7, Table 3.
access_number	Carries the Access Number element of the data header as specified in EN 13757-3:2004, 5.8. It has unsigned binary coding, and it is incremented (modulo 256) by one before or after each RSP-UD from the slave. Since it can also be used to enable private end users to detect an unwanted over-frequently readout of their consumption meters, it should not be resettable by any bus communication.
status	Carries the Status byte element of the data header as specified in EN 13757-3:2004, 5.9, Tables 4 and 5.
alarm	Carries the Alarm state specified in EN 13757-3:2004, Annex D. It is coded with data type D (Boolean, in this case 8 bit). Set bits signal alarm bits or alarm codes. The meaning of these bits is manufacturer specific.
Method description	
slave_install (data)	Installs a slave device, which is unconfigured yet (its primary address is 0). This method can be successfully invoked only if the value of the

	primary_address attribute is 0. The following actions are performed: • the M-Bus address 0 is checked for presence of a new device; • if no uninstalled M-Bus slave is found, the method invocation fails; • if the slave_install method is invoked with no parameter, then the primary address is assigned automatically. This is done by checking the primary_address attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The primary_address attribute is set to this address and it is then transferred to the M-Bus slave device; • if the slave_install method is invoked with a primary address as a parameter, then the primary_address attribute is set to this value and it is then transferred to the M-Bus slave device. data::= unsigned (no data, or a valid primary address) NOTE 3 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2004, E.5.
slave_deinstall (data)	De-installs the slave device. The main purpose of this service is to uninstall the M-Bus slave device and to prepare the master for the installation of a new device. The following actions are performed: • the M-Bus address is set to 0 in the M-Bus slave device; • the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected; • the encryption status attribute shall be reset to 0; • the attribute primary_address is also set to 0. NOTE 4 A new M-Bus slave can be installed only once the value of the primary_address attribute is 0. data::= integer (0)
capture	Capture values – as specified by the capture_definition attribute – from the M-Bus slave device. data::= integer (0)
reset_alarm	Reset alarm state of the M-Bus slave device. data::= integer (0)
synchronize_clock	Synchronize the clock of the M-Bus slave device with that of the M-Bus client device. data::= integer (0)
data_send	Send data to the M-Bus slave device. data array data_definition_element

	data_definition_element::= structure
	
	{
	data_information_block: octet-string,
	value_information_block : octet-string
	data: CHOICE
	{
	-- simple data types
	null-data [0],
	bit-string [4],
	double-long [5],
	double-long-unsigned [6],
	octet-string [9],
	visible-string [10],
	UTF8-string [12],
	bcd [13],
	integer [15],
	long [16],
	unsigned [17],
	long-unsigned [18],
	long64 [20],
	long64-unsigned [21],
	float32 [23],
	float64 [24]
	}
	}

set_encryption_key	Sets encryption key to be used with the M-Bus slave device. Changing the encryption key requires two steps: first, the key is sent to the M-Bus slave, encrypted with the default key, using the transfer_key method. Second, the key is set in the M-Bus master using the set_encryption_key method. data::= octet-string (encryption_key) After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus telegrams is disabled. Encryption can be disabled by invoking the set_encryption_key method with null data as a parameter.
transfer_key	Transfers an encryption key to the M-Bus slave. data::= octet-string (encrypted_key) Each M-Bus slave device shall be delivered with a default encryption key. Before encrypted M-Bus telegrams can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the transfer_key method. The method invocation parameter is the operational encryption key encrypted with the default key of the M-Bus slave device. The M-Bus telegram sent is not encrypted. After the execution, the encryption is enabled and all further telegrams are encrypted. A new encryption key can be set in the M-Bus client by invoking the set_encryption_key method with the new encryption key as a

parameter.

With further invocations of the transfer_key method, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus telegram is encrypted.

When an M-Bus slave is de-installed, the encryption key is destroyed, but the default key is not affected. Encryption remains disabled until a new encryption is transferred.

5.6.3 Wireless Mode Q channel (class_id: 73, version: 1)

Instances of this IC define the operational parameters for communication using the mode Q interfaces. See also EN 13757-5.

Wireless Mode Q channel		0...n	class_id = 73, version = 1			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. addr_state	(static)	enum				x + 0x08
3. device_address	(static)	octet-string				x + 0x10
4. address_mask	(static)	octet-string				x + 0x18
Specific methods		***m/o***				

Attribute description

addr_state Defines whether or not the device has been assigned an address since last power up of the device

enum: (0) not assigned an address yet,
(1) assigned an address either by manual setting or by automated method

device_address The currently assigned address of the device on the network

address_mask The group address the device will respond to when short form addressing is used

5.6.4 M-Bus master port setup (class_id: 74, version: 0)

Instances of this IC define the operational parameters for communication using the EN 13757-2 interfaces if the device acts as an M-bus master.

M-Bus master port setup		0...n	class_id = 74, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	7	3	x + 0x08
Specific methods		***m/o***				

Attribute description

logical_name Identifies the “M-Bus master port setup” object instance. See 6.2.18.

comm_speed	The communication speed supported by the port enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud

5.7 Interface classes for setting up data exchange over the Internet

5.7.1 TCP-UDP setup (class_id: 41, version: 0)

This IC allows modelling the setup of the TCP or UDP sub-layer of the COSEM TCP or UDP based transport layer of a TCP-UDP/IP based communication profile.

In TCP-UDP/IP based communication profiles, all AAs between a physical device hosting one or more COSEM client application processes and a physical device hosting one or more COSEM server APs rely on a single TCP or UDP connection. The TCP or UDP entity is wrapped in the COSEM TCP-UDP based transport layer. Within a physical device, each AP – client AP or server logical device – is bound to a Wrapper Port (WPort). The binding is done with the help of the SAP Assignment object. See 5.3.3.

On the other hand, a COSEM TCP or UDP based transport layer may be capable to support more than one TCP or UDP connections, between a physical device and several peer physical devices hosting COSEM APs.

When a COSEM physical device supports various data link layers – for example Ethernet and PPP – an instance of the TCP-UDP setup object is necessary for each of them.

TCP-UDP setup		**0...n**	**class_id = 41, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. TCP-UDP_port	(static)	long-unsigned				x + 0x08
3. IP_reference	(static)	octet-string				x + 0x10
4. MSS	(static)	long-unsigned	40	65 535	576	x + 0x18
5. nb_of_sim_conn	(static)	unsigned	1			x + 0x20
6. inactivity_time_out	(static)	long-unsigned			180	x + 0x28
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the “TCP-UDP setup” object instance. See 6.2.19.
TCP-UDP_port	Holds the TCP-UDP port number on which the physical device is listening for the DLMS/COSEM application. For DLMS/COSEM, the following port numbers have been registered by the IANA. See http://www.iana.org/assignments/port-numbers (last update 2010-08-17) dlms/cosem 4059/TCP DLMS/COSEM dlms/cosem 4059/UDP DLMS/COSEM

IP_reference	References an IP setup object by its logical name. The referenced object contains information about the IP Address settings of the IP layer supporting the TCP-UDP layer.
MSS	With the help of the Maximum Segment Size (MSS) option, a TCP layer entity can indicate the maximum receive segment size to its partner. Note, that: • this option shall only be sent in the initial connection request (i.e. in segments with the SYN control bit sent); • if this option is not present, conventionally MSS is considered as its default value, 576; • MSS is not negotiable; its value is indicated by this attribute.
nb_of_sim_conn	The maximum number of simultaneous connections the COSEM TCP-UDP based transport layer is able to support.
inactivity_time_out	Defines the time, expressed in seconds over which, if no frame is received from the COSEM client, the inactive TCP connection shall be aborted. When this value is set to 0, this means that the inactivity_time_out is not operational. In other words, a TCP connection, once established, in normal conditions – no power failure, etc. – will never be aborted by the COSEM server. Note, that all actions related to the management of the inactivity time-out function – measuring the inactivity time, aborting the TCP connection if the time-out is over, etc. – are managed inside the TCP-UDP layer implementation.

5.7.2 IPv4 setup (class_id: 42, version: 0)

This IC allows modelling the setup of the IPv4 layer, handling all information related to the IP Address settings associated to a given device and to a lower layer connection on which these settings are used.

There shall be an instance of this IC in a device for each different network interface implemented. For example, if a device has two interfaces (using the TCP/IP profile on both of them), there shall be two instances of the IPv4 setup IC in that device: one for each of these interfaces.

IPv4 setup	0...n	class_id = 42, version = 0			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. DL_reference (static)	octet-string				x + 0x08
3. IP_address	double-long-unsigned				x + 0x10
4. multicast_IP_address	array				x + 0x18
5. IP_options	array				x + 0x20
6. subnet_mask	double-long-unsigned				x + 0x28
7. gateway_IP_address	double-long-unsigned				x + 0x30
8. use_DHCP_flag (static)	boolean				x + 0x38
9. primary_DNS_address	double-long-unsigned				x + 0x40
10. secondary_DNS_address	double-long-unsigned				x + 0x48
Specific methods	***m/o***				
1. add_mc_IP_address (data)	o				x + 0x60
2. delete_mc_IP_address (data)	o				x + 0x68
3. get_nbof_mc_IP_addresses (data)	o				x + 0x70

Attribute description

logical_name

Identifies the "IPv4 setup" object instance. See 6.2.19.

DL_reference

References a Data link layer (e.g. Ethernet or PPP) setup object by its logical name. The referenced object contains information about the specific settings of the data link layer supporting the IP layer.

IP_address

Carries the value of the IP address (IPv4) of this physical device on the network to which the device is connected via the referenced lower layer interface.

It can be either (static) or (dynamic). In the latter case, dynamic IP address assignment (for example DHCP) is used.

If no IP address is assigned, the value is 0.

multicast_IP_address

Contains an array of IP addresses. IP addresses in this array shall fall into the multicast group address range ("Class D" addresses, including IP addresses in the range of 224.0.0.0 to 239.255.255.255).

When a device receives an IP datagram with one of these IP addresses in the destination IP address field, it shall consider that this datagram is addressed to it.

multicast_IP_address::= array double-long-unsigned

IP_options

Contains the necessary parameters to support the selected IP options – for example Datagram time-stamping or security services (IPSec).

IP_options::= array IP_options_element

IP_options_element::= SEQUENCE

{
IP-Option-Type: unsigned,
IP-Option-Length: unsigned,
IP-Option-Data: octet-string
}

Allowed IP-Option-Types:

- Security – 0x82

If this option is present, the device shall be allowed to send security, compartmentation, handling restrictions and TCC (closed user group) parameters within its IP Datagrams. The value of the IP-Option-Length Field shall be 11, and the IP-Option-Data shall contain the value of the Security, Compartments, Handling Restrictions and Transmission Control Code values, as specified in STD 0005 / RFC 791.

– Loose Source and Record Route – 0x83

If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The IP-Option-length and IP-Option-Data values are specified in STD 0005 / RFC 791.

– Strict Source and Record Route – 0x89

If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The IP-Option-length and IP-Option-Data values are specified in STD 0005 / RFC 791.

– Record Route – 0x07

If this option is present, the device shall as well:

- send originated IP Datagrams with that option, providing means to record the route of these Datagrams;
- as a router, send routed IP Datagrams with the route option adjusted according to this option.

The IP-Option-length and IP-Option-Data values are specified in STD 0005 / RFC 791.

– Internet Timestamp – 0x44

If this option is present, the device shall as well:

- send originated IP Datagrams with that option, providing means to time-stamp the datagram in the route to its destination;
- as a router, send routed IP Datagrams with the time-stamp option

	adjusted according to this option. The IP-Option-length and IP-Option-Data values are specified in STD 0005 / RFC 791.
subnet_mask	Contains the subnet mask. When sub-networking is used in a network segment, each device concerned shall behave conforming to the sub-networking rules. In order to do that, the device, besides of its IP address, needs also to know how the IP address is structured within this sub-networked segment. The subnet_mask attribute carries this information. With IPv4, the subnet_mask is a 32 bits word, expressed exactly in the same format as an IP Address (for example 255.255.255.0), but has another meaning: the '0' bits of the subnet_mask indicate the portion of the IP Address which is still used as Device_ID on a sub-networked IP Network[7].
gateway_IP_address	Contains the IP Address of the gateway device. In most IP implementations, there is a code in the module that handles outgoing datagrams to decide if a datagram can be sent directly to the destination on the local network or if it shall be sent to a gateway. In order to be able to send non-local datagrams to the gateway, the device shall know the IP address of the gateway device assigned to the given network segment. If no IP address is assigned, the value is 0.
use_DHCP_flag	When this flag is set to TRUE, the device uses DHCP (Dynamic Host Configuration Protocol) to dynamically determine the IP_address, subnet_mask and gateway_IP_address parameters. On the other hand, when this flag is set to FALSE, the IP_address, subnet_mask and gateway_IP_address parameters shall be locally set.
primary_DNS_ address	The IP Address of the primary Domain Name Server (DNS). If no IP address is assigned, the value is 0.
secondary_DNS_ address	The IP Address of the secondary Domain Name Server (DNS). If no IP address is assigned, the value is 0.
Method description	
add_mc_IP_address (IP_Address)	Adds one multicast IP address to the multicast_IP_Address array. IP_Address::= double-long-unsigned

[7] See more about sub-networking in RFC 940 and RFC 950.

delete_mc_IP_ address (IP_Address)	Deletes one IP Address from the multicast_IP_Address array. The IP Address to be deleted is identified by its value. IP_Address::= double-long-unsigned
get_nbof_mc_IP_ addresses (data)	Returns the number of IP Addresses contained in the multicast_IP_Address array. data::= unsigned

5.7.3 MAC address setup (class_id: 43, version: 0)

NOTE 1 The name and the use of this interface class has been changed from "Ethernet setup" to "MAC address setup" to allow a more general use, without changing the version.

Instances of this IC hold the MAC address of the physical device. (Or, more generally, a device or software). There shall be an instance of this IC for each network interface of a physical device.

NOTE 2 In the case of the three-layer HDLC based communication profile, the MAC address (lower HDLC address) is carried by an IEC HDLC setup object.

NOTE 3 In the case of the S-FSK PLC communication profile, the MAC address is carried by a S-FSK Phy&MAC setup object.

MAC address setup		**0...n**	**class_id = 43, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. MAC_address		octet-string				x + 0x08
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the "MAC address setup" object instance. See 6.2.19.
MAC_address	Holds the MAC address.

5.7.4 PPP setup (class_id: 44, version: 0)

This IC allows modelling the setup of interfaces using the PPP protocol, by handling all information related to PPP settings associated to a given physical device and to a lower layer connection on which these settings are used. There shall be an instance of this IC for each network interface of a physical device.

PPP setup		**0...n**	**class_id = 44, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. PHY_reference	(static)	octet-string				x + 0x08
3. LCP_options	(static)	LCP_options				x + 0x10
4. IPCP_options	(static)	IPCP_options				x + 0x18
5. PPP_authentication	(static)	PPP_auth				x + 0x20
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the “PPP setup” object instance. See 6.2.19.
PHY_reference	References another object by its logical_name. The object referenced contains information about the specific physical layer interface, supporting the PPP layer.
LCP_options	This attribute contains the parameters for the Link Control Protocol options. These options include: • Maximum-Receive-Unit (MRU, Type 1, STD 0051 / RFC 1661). This configuration option may be sent to inform the peer that the implementation can receive larger packets, or to request that peer send smaller packets. The default value is 1 500 octets; • Async-Control-Character_Map (ACCM, Type 2, STD 0051 / RFC 1662): This configuration option provides a method to negotiate the use of control character transparency on asynchronous links; • Authentication-Protocol (Type 3, STD 0051 / RFC 1661, PAP, CHAP or EAP); This configuration option provides a method to negotiate the use of a specific protocol for authentication. By default, authentication is not required; • Magic-Number (Type 5, STD 0051 / RFC 1661). This configuration option provides a method to detect looped-back links and other data link layer anomalies; • Protocol-Field-Compression (PFC, Type 7, STD 0051 / RFC 1661). This configuration option provides a method to negotiate the compression of the PPP protocol field; • Address-and-Control-Field-Compression (ACFC, Type 8, STD 0051 / RFC 1661). This configuration option provides a method to negotiate the compression of the data link layer address and control fields; • FCS-Alternatives (Type 9, RFC 1570). This configuration option provides a method for an implementation to specify another FCS format to be sent by the peer, or to negotiate away the FCS altogether; • Call-back (Type 13, RFC 1570). This configuration option provides a method for an implementation to request a dial-up peer to call back. This provides enhanced security by ensuring that the remote site can connect only from a single location as defined by the call-back number. The structure of this attribute is the following: LCP_options::= SEQUENCE OF LCP_options_element LCP_options_element ::= SEQUENCE { LCP-option-type: unsigned, LCP-option-length: unsigned, LCP-option-data: CHOICE { MRU [1] long-unsigned, ACCM [2] double-long-unsigned, Auth-Prot [3] long-unsigned, Mag-Num [5] double-long-unsigned, ProtF-Compr [7] boolean, AdCtr-Compr [8] boolean,

```
                    FCS-Alter       [9] unsigned,
                    Callback        [13] callback-data
                    }
}
LCP_option_type::= enum
{
        Max-Rec-Unit                        (1),
        Async-Control-Char-Map              (2),
        Auth-Protocol                       (3),
        Magic-Number                        (5),
        Protocol-Field-Compression          (7),
        Address-and-Ctr-Compression         (8),
        FCS-Alternatives                    (9),
        Callback                            (13)
}
```

For the LCP-option-data element, the following applies:

The default value of MRU is 1 500.

The value of the Auth-Prot (Authentication Protocol) element indicates the authentication protocol used on the given PPP link.

Possible values today are:

0x0000 – No authentication protocol is used,
0xc023 – The PAP protocol is used,
0xc223 – The CHAP protocol is used,
0xc227 – The EAP protocol is used.

The value of the FCS-Alter (FCS Alternatives) options field identifies the FCS used. These are assigned as follows:

bit 1 Null FCS,
bit 2 CCITT 16-bit FCS,
bit 4 CCITT 32-bit FCS

```
Callback-data::= SEQUENCE
{
        callback-active         boolean,    // default: false,
        callback-data-length    unsigned,
        callback-operation      unsigned,
        callback-message        octet-string
}
```

The callback-active member indicates whether the callback option is active on this PPP link.

```
callback-operation::= enum
{
        Location-is-determined-by-user-authentication       (0),
        Dialling-string                                     (1),
        Location-identifier                                 (2),
        E.164-number                                        (3),
        X500-distinguished-name                             (4),
        Location-is-determined-during-CBCP-negotiation      (6)
}
```

The callback-message field is zero or more octets, and its general contents are determined by the callback-operation field. The actual

format of the information is site or application specific (see in RFC 1570).

NOTE 1 For more details on Link Control Protocol, please refer to RFC 1661.

NOTE 2 For latest assigned numbers, see RFC 1700.

IPCP_options

This attribute contains the parameters for the IP Control Protocol – the Network Control Protocol module of the PPP for negotiating IP parameters on the PPP link options. These options include:

- IP-Compression-Protocol (Type 2, RFC 1332). This parameter indicates the IP compression protocol supported within the PPP link described by this object;
- Preferred-Local-IP-Address (Type 3, RFC 1332). This configuration option provides a way to negotiate the IP address to be used on the local end of the link. It allows the sender of the Configure-Request to state, which IP-address is desired, or to request that the peer provide the information. The peer can provide this information by NAK-ing the option, and returning a valid IP-Address;
- Preferred-Peer-IP-Addresses. This configuration option provides a way to negotiate the IP Address to be used on the remote end of the link. When the Grant-Access-Only-to-Pref-Peer-on-List parameter is set to TRUE, the device shall accept the PPP connection only with a remote device having one of the IP Addresses on this list. When the Use-Static-IP-Pool parameter is set to TRUE, the COSEM Server device shall try to assign one of these IP Addresses to the remote device;
- Grant-Access-Only-to-Pref-Peer-on-List (GAO). This parameter indicates whether the device can accept PPP connection only with peer devices with IP Address on the above list or not. Its default value is FALSE;
- Use-Static-IP-Pool (USIP). This parameter indicates whether the device should try to assign one of the IP Addresses of the Preferred-Peer-IP-Addresses to the remote end device during the IP Address negotiation phase or not.

The structure of this attribute is as follows:

```
IPCP_options::= SEQUENCE OF IPCP_options_element
IPCP_options_element::= SEQUENCE
{
        IPCP-option-type        unsigned,
        IPCP-option-length      unsigned,
        IPCP-option-data        CHOICE
        {
        IP-Comp-Prot    [2] long-unsigned,
        Pref-Local-IP   [3] double-long-unsigned,
        Pref-Peer-IP    [20] SEQUENCE OF double-long-unsigned,
        GAO             [21] boolean,
        USIP            [22] boolean
        }
}
IPCP-option-type::= enum
{
        IP-Comp-Prot    (2),
        Pref-Local-IP   (3),
        Pref-Peer-IP    (20),
        GAO             (21),
```

```
        USIP            (22)
}
```

Possible values for the IP-Compression-Protocol (IP-Comp-Prot) parameter today are:

0x0000 – No IP Compression is used (default),
0x002d – Van Jacobson (RFC 1332),
0x0061 – IP Header Compression (RFC 2507, 3544)
0x0003 – Robust Header Compression (ROHC, RFC 3241)

NOTE 3 For more details on IPCP, please refer to RFC 1332.

NOTE 4 For latest assigned numbers, see RFC 1700.

PPP_ authentication

Contains the parameters required by the PPP authentication procedure used.

```
PPP_authentication::= CHOICE

{
        No-authentication:      [0] NULL,
        PAP-login               [1] structure,
                {
                        user-name:      octet-string,
                        PAP-password:   octet-string
                }
        CHAP-algorithm  [2] structure,
                {
                        user-name:      octet-string,
                        algorithm_id:           unsigned
                                -- default: 5 (MD5)
                }
        EAP-params:             [3] supported-EAP-types
}
```

Possible values for CHAP-algorithm-id parameter today are as follows:

0x05 – CHAP with MD5 (default),
0x06 – SHA-1,
0x80 – MS-CHAP,
0x81 – MS-CHAP-2

New values can be used as they become assigned.

When CHAP is used, a “secret” is also required to verify the “challenge”[8] sent by the Client. This “secret” is not accessible in the PPP setup object.

```
supported-EAP-types::= SEQUENCE

{
        md5-challenge           boolean,
        one-time-password       boolean,
        generic-token-card      boolean
}
```

5.7.5 GPRS modem setup (class_id: 45, version: 0)

This IC allows setting up GPRS modems, by handling all data necessary data for modem management.

8 For more details about CHAP, please refer to RFC 1994.

GPRS modem setup		0...n	class_id = 45, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. APN	(static)	octet-string				x + 0x08
3. PIN_code	(static)	long-unsigned				x + 0x10
4. quality_of_service	(static)	structure				x + 0x18
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the "GPRS modem setup" object instance. See 6.2.19.
APN	Defines the access point name of the network.
PIN_code	Holds the personal identification number.

quality_of_service	Specifies the quality of service parameters. It is a structure of two elements:
	• the first one defines the default or minimum characteristics of the concerned network. These parameters have to be set to best effort value;
	• the second element defines the requested parameters.
	quality_of_service::= structure
	{
	default: qos_element,
	requested: qos_element
	}
	qos_element::= structure
	{
	precedence: unsigned,
	delay: unsigned,
	reliability: unsigned,
	peak throughput: unsigned,
	mean throughput: unsigned
	}

5.7.6 SMTP setup (class_id: 46, version: 0)

This IC allows setting up data exchange using the SMTP protocol.

SMTP modem setup		0...n	class_id = 46, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. server_port	(static)	long-unsigned			25	x + 0x08
3. user_name	(static)	octet-string				x + 0x10
4. login_password	(static)	octet-string				x + 0x18
5. server_address	(static)	octet-string				x + 0x20
6. sender_address	(static)	octet-string				x + 0x28
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the “SMTP setup” object instance. See 6.2.19.
server_port	Defines the value of the TCP-UDP port related to this protocol. By default, this value is the SMTP port numberID assigned by IANA: smtp 25/tcp Simple Mail Transfer smtp 25/udp Simple Mail Transfer
user_name	Defines the user name to be used for the login to the SMTP server.
login_password	Password to be used for login. When the string is void, this means that there is no authentication.
server_address	Defines the server address as an octet string. This server address can be a name, which shall be resolvable by the primary DNS or the secondary DNS. In the case when it is directly the IP address of the server, which is specified here, it shall be a string in dotted format. EXAMPLE 163.187.45.87.
sender_address	Defines the sender address as an octet string. This sender address can be a name. In the case when it is directly the IP address of the sender, which is specified here, it will be a string in dotted format.

5.8 Interface classes for setting up data exchange using S-FSK PLC

5.8.1 General

This subclause specifies COSEM interface classes to set up and manage the protocol layers of DLMS/COSEM PLC communication profiles:

- the S-FSK Physical layer and the MAC sub-layer as defined in IEC 61334-5-1 and IEC 61334-4-512

NOTE 1 ICs for setting up other PLC lower layer profiles may be added later.

- the LLC sub-layer as specified in IEC 61334-4-32;
- the LLC sub-layer as specified in ISO/IEC 8802-2.

The MIB variables / logical link parameters specified in IEC 61334-4-512, IEC 61334-5-1, IEC 61334-4-32 and ISO/IEC 8802-2 respectively have been mapped to attributes and/or methods of COSEM ICs. The specification of these elements has been taken from the above-mentioned standards and the text has been adapted to the DLMS/COSEM environment.

NOTE 2 IEC 61334-4-512 also specifies some management variables to be used on the Client side. However, the Client side object model is not covered in this standard.

5.8.2 Definitions and abbreviations related to the S-FSK PLC profile

5.8.2.1 Definitions

For the purposes of this document, the terms and definitions given in IEC/TR 62051 and IEC/TR 62051-1 and the following apply.

5.8.2.1.1
initiator
user-element of a client System Management Application Entity (SMAE)

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

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

5.8.2.1.2
active initiator
initiator which issues or has last issued a CIASE Register request when the server is in the unconfigured state

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

5.8.2.1.3
new system
server system which is in the unconfigured state: its MAC address equals "NEW-address"

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

5.8.2.1.4
new system title
system-title of a new system

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

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

5.8.2.1.5
registered system
server system which has an individual valid MAC address (therefore, different from "NEW Address", see IEC 61334-5-1: Medium Access Control)

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

5.8.2.1.6
reporting system
server system which issues a DiscoverReport

[SOURCE: IEC 61334-4-511:2000 3.9.6, modified]

5.8.2.1.7
sub-slot
time needed to transmit two bytes by the physical layer

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

5.8.2.1.8
timeslot
time needed to transmit a physical frame

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

5.8.2.2 Abbreviations

Abbreviation	Explanation
CC	Current Credit (in the case of the S-FSK PLC profile)
DC	Delta credit
IC	Initial credit
CIASE	Configuration Initiation Application Service Element
FIFO	First-In-First-Out
LLC	Logical Link Control

Abbreviation	Explanation
MAC	Medium Access Control
MIB	Management Information Base
UNC	Unconfigured (server, in the case of the S-FSK PLC profile)

5.8.3 Overview

COSEM objects for setting up the PLC channel and the LLC layer, if implemented, shall be located in the Management Logical Device of COSEM servers.

Figure 15 shows an example with a COSEM physical device comprising three logical devices.

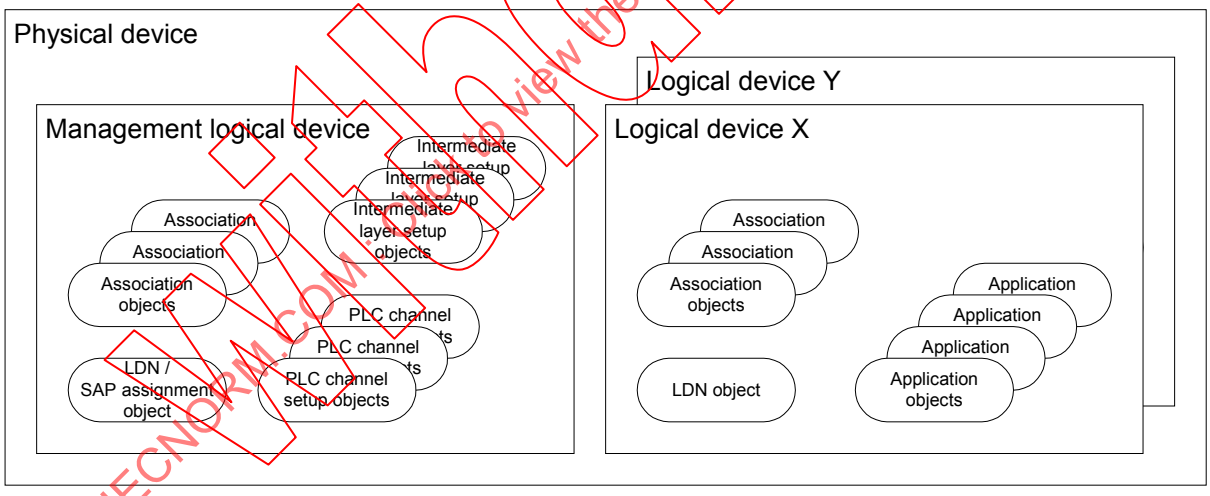
Each logical device shall contain a Logical Device Name (LDN) object.

NOTE As in this example there is more than one logical device, the mandatory Management Logical Device contains a SAP Assignment object instead of a Logical Device Name object.

Each logical device contains one or more Association objects, one for each client supported.

The management logical device contains the setup objects of the physical and MAC layers of the PLC channel, as well as setup objects for the intermediate layer(s). It may contain further application objects.

The other logical devices, in addition to the Association and Logical Device Name objects mentioned above, contain further application objects, holding parameters and measurement values.



IEC 1110/13

Figure 15 – Object model of DLMS/COSEM servers

IEC 61334-4-512 uses DLMS named variables to model the MIB objects and specifies their DLMS name in the range 8...184. For compatibility with existing implementations, the short names 8...400 [sic] are reserved for devices using the IEC 61334-5-1 S-FSK PLC profiles without COSEM. Therefore, when mapping the attributes and methods of the COSEM objects specified in this standard to DLMS named variables (SN mapping) this range shall not be used.

Table 10 – Mapping IEC 61334-4-512 MIB variables to COSEM IC attributes / methods

Name	Reference (unless otherwise indicated)	Interface class	class_id / attribute / method
S-FSK Physical layer management			
delta-electrical-phase	variable 1	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	50 / Attr. 3
max-receiving-gain	variable 2		50 / Attr. 4
max-transmitting-gain	–		50 / Attr. 5
search-initiator-threshold	–		50 / Attr. 6
frequencies	–		50 / Attr. 7
transmission-speed	–		50 / Attr. 15
MAC layer management			
mac-address	variable 3	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	50 / Attr. 8
mac-group-addresses	variable 4		50 / Attr. 9
repeater	variable 5		50 / Attr. 10
repeater-status	–		50 / Attr. 11
search-initiator time-out	–	S-FSK MAC synchronization timeouts (class_id: 52, version: 0)	52 / Attr. 2
synchronization-confirmation-time-out	variable 6		52 / Attr. 3
time-out-not-addressed	variable 7		52 / Attr. 4
time-out-frame-not-OK	variable 8		52 / Attr. 5
min-delta-credit	variable 9	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	50 / Attr. 12
initiator-mac-address	IEC 61334-5-1:2001, 4.3.7.6		50 / Attr. 13
synchronization-locked	variable 10		50 / Attr. 14
IEC 61334-4-32 LLC layer management			
max-frame-length	IEC 61334-4-32:1996, 5.1.4	IEC 61334-4-32 LLC setup (class_id: 55, version: 1)	55 / Attr. 2
reply-status-list	variable 11		55 / Attr. 3
broadcast-list	variable 12	–	–
L-SAP-list	variable 13	NOTE In DLMS/COSEM, L-SAPs of logical devices are held by a SAP Assignment object	
ACSE management			
application-context-list	variable 14	NOTE In DLMS/COSEM the Association objects play a similar role.	
Application management			
Active-initiator	variable 15	S-FSK Active initiator (class_id: 51, version: 0)	51 / Attr. 2
MIB system objects			
reporting-system-list	variable 16	S-FSK Reporting system list (class_id: 56, version: 0)	56 / Attr. 2
Other MIB objects			
Reset-NEW-not-synchronized	variable 17	S-FSK Active initiator (class_id: 51, version: 0)	51 / Method 1
new-synchronization	IEC 61334-5-1:2001, 4.3.7.6	–	
initiator-electrical-phase	variable 18		50 / Attr. 2
broadcast-frames-counter	variable 19	S-FSK MAC counters (class_id: 53, version: 0)	53 / Attr. 4
repetitions-counter	variable 20		53 / Attr. 5
transmissions-counter	variable 21		53 / Attr. 6
CRC-OK-frames-counter	variable 22		53 / Attr. 7
CRC-NOK-frames-counter	–		53 / attr. 8
synchronization-register	variable 23		53 / Attr. 2
desynchronization-listing	variable 24		53 / Attr. 3

Table 10 above shows the mapping of MIB variables to attributes and/or methods of COSEM ICs.

Note, that on one hand, not all MIB variables specified in IEC 61334-4-512 have been mapped to attributes and methods of COSEM ICs. On the other hand, some new management variables are specified in this standard.

5.8.4 S-FSK Phy&MAC set-up (class_id: 50, version: 1)

NOTE 1 The use of version 0 of this interface class is deprecated.

An instance of the "S-FSK Phy&MAC set-up" class stores the data necessary to set up and manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

S-FSK Phy&MAC setup		**0...n**	**class_id = 50, version = 1**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. initiator_electrical_phase	(static)	enum	0	3		x + 0x08
3. delta_electrical_phase	(dyn.)	enum	0	6		x + 0x10
4. max_receiving_gain	(static)	unsigned				x + 0x18
5. max_transmitting_gain	(static)	unsigned				x + 0x20
6. search_initiator_threshold	(static)	unsigned			98	x + 0x28
7. frequencies	(static)	frequencies_type				x + 0x30
8. mac_address	(dyn.)	long-unsigned			FFE	x + 0x38
9. mac_group_addresses	(static)	array				x + 0x40
10. repeater	(static)	enum				x + 0x48
11. repeater_status	(dyn.)	boolean				x + 0x50
12. min_delta_credit	(dyn.)	unsigned				x + 0x58
13. initiator_mac_address	(dyn.)	long-unsigned				x + 0x60
14. synchronization_locked	(dyn.)	boolean				x + 0x68
15. transmission_speed	(static)	enum	0	6	3	x + 0x70
Specific methods		***m/o***				

Attribute description

logical_name Identifies the "S-FSK Phy&MAC setup" object instance. See 6.2.20

initiator_electrical_phase Holds the MIB variable initiator-electrical-phase (variable 18) specified in IEC 61334-4-512:2001, 5.8.

It is written by the client system to indicate the phase to which it is connected.

enum: (0) Not defined (default),
(1) Phase 1,
(2) Phase 2,
(3) Phase 3

NOTE 2 This enumeration is different from that of IEC 61334-4-512.

delta_electrical_phase Holds the MIB variable delta-electrical-phase (variable 1) specified

	in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1:2001, 3.5.5.3. It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined: enum: (0) Not defined: the server is temporarily not able to determine the phase difference, (1) The server system is connected to the same phase as the client system. The phase difference between the server's connecting phase and the client's connecting phase is equal to: (2) 60 degrees, (3) 120 degrees, (4) 180 degrees, (5) -120 degrees, (6) -60 degrees
max_receiving_gain	Holds the MIB variable max-receiving-gain (variable 2) specified in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1, 3.5.5.3. Corresponds to the maximum allowed gain bound to be used by the server system in the receiving mode. The default unit is dB. NOTE 3 In IEC 61334-4-512, no units are specified. The possible values of the gain can depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
max_transmitting_gain	Holds the value of the max-transmitting-gain. Corresponds to the maximum attenuation bound to be used by the server system in the transmitting mode. The default unit is dB. The possible values of the gain can depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
search_initiator_ threshold	This attribute is used in the intelligent search initiator process. If the value of the initiator signal is above the value of this attribute, a fast synchronization process is possible. The default value is 98 dBµV.

frequencies	Contains frequencies required for S-FSK modulation.
	
	frequencies_type::= structure
	{
	mark_frequency: double-long-unsigned,
	space_frequency: double-long-unsigned
	}
	The default unit is Hz.

mac_address	Holds the MIB variable mac-address (variable 3) specified in

	IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 MAC addresses are expressed on 12 bits. Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is “NEW-address”. This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address". This attribute is set to NEW: • by the MAC sub-layer, once the time-out-not-addressed delay is exceeded; • when a client system “resets” the server system. See 5.8.5. When this attribute is set to NEW: • the system loses its synchronization (function of the MAC-sublayer); • the mac_group_address attribute is reset (array of 0 elements); • the system automatically releases all AAs which can be released. NOTE 5 The second item is not present in IEC 61334-4-512. The predefined MAC addresses are shown in Table 11.
mac_group_addresses	Holds the MIB variable mac-group-address (variable 4) specified in in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Contains a set of MAC group addresses used for broadcast purposes. array mac-address mac-address::= long-unsigned The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included in this list. These ones are internal predefined values. This attribute shall be written by the initiator using DLMS services to declare specific MAC group addresses on a server system. This attribute is locally read by the MAC sublayer when checking the destination address field of a MAC frame not recognized as an individual address or as one of the three predefined values (ALL-configured-address, ALL-physical-address and NO-BODY).
repeater	Holds the MIB variable repeater (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. It specifies whether the server system effectively repeats all frames or not.

	enum: (0) never repeater, (1) always repeater, (2) dynamic repeater If the repeater variable is equal to 0, the server system should never repeat the frames. If it is set to 1, the server system is a repeater: it has to repeat all frames received without error and with a current credit greater than zero. If it is set to 2, then the repeater status can be dynamically changed by the server itself. NOTE 6 Value 2 is not specified in IEC 61334-4-512. This attribute is internally read by the MAC sub-layer each time a frame is received. The default value shall be specified in companion specifications.
repeater_status	Holds the current repeater status of the device. boolean: FALSE = no repeater, TRUE = repeater
min_delta_credit	Holds the MIB variable min-delta-credit (variable 9) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 7 Only the three least significant bits are used. Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields of a correct received MAC frame. The delta-credit minimum value of a correct received MAC frame, directed to a server system, is held by this variable. The default value is set to the maximal initial credit (see IEC 61334-5-1:2001, 4.2.3.1 for further explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can reinitialise this variable by setting its value to the maximal initial credit.
initiator_mac_address	Holds the MIB variable initiator-mac-address specified in IEC 61334-5-1:2001, 4.3.7.6. Its value is either the MAC address of the active-initiator or the NO-BODY address, depending on the value of the synchronization_locked attribute (see below). See also IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2.
synchronization_locked	Holds the MIB variable synchronization-locked (variable 10) specified in IEC 61334-4-512:2001, 5.3. Controls the synchronization locked / unlocked state. See IEC 61334-5-1 for more details. If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state. In this state, the initiator-mac-address is always equal to the MAC address field of the active-

	initiator MIB object. See attribute 2 of the S-FSK Active initiator IC in 5.8.5. If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the initiator_mac_address attribute is always set to the NO-BODY value: a value change in the MAC address field of the active-initiator MIB object does not affect the content of the initiator_mac_address attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications. NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of which is equal to the value of the initiator_mac_address attribute.
transmission_speed	The transmission speed supported by the physical device. See also IEC 61334-5-1:2001, 3.2.2. enum: (0) 300 baud 50Hz; (1) 600 baud 60 Hz; (2) 1 200 baud 720 baud; (3) -- default 2 400 baud 1 440 baud; (4) 4 800 baud 2 880 baud; (5) 7, 200 baud 5 760 baud; (6) 9 600 baud 8 640 baud; 11 520 baud

Table 11 – MAC addresses in the S-FSK profile

Address	Value
NO-BODY	000
Local MAC	001…FIMA-1
Initiator	FIMA…LIMA
MAC group address	LIMA + 1…FFB
All-configured	FFC
NEW	FFE
All Physical	FFF
NOTE MAC addresses are expressed on 12 bits. These addresses are specified in IEC 61334-5-1:2001, 4.2.3.2, 4.3.7.5.1, 4.3.7.5.2 and 4.3.7.5.3. FIMA = First Initiator MAC address; C00 LIMA = Last Initiator MAC address; DFF	

5.8.5 S-FSK Active initiator (class_id: 51, version: 0)

An instance of the “S-FSK Active initiator” IC stores the data of the active initiator. The active initiator is the client system, which has last registered the server system with a CIASE Register request. IEC 61334-4-511:2000, 7.2 applies.

S-FSK Active initiator		0...n	class_id = 51, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. active_initiator	(dyn.)	initiator_descriptor				x + 0x08
Specific methods		***m/o***				
1. reset_NEW_not_synchronized						x + 0x10

Attribute description

logical_name

Identifies the "S-FSK Active initiator" object instance. For logical names, see 6.2.20.

active_initiator

Holds the MIB variable *active-initiator* (variable 15) specified in IEC 61334-4-512:2001, 5.6.

Contains the identifiers of the active initiator, which has last registered the system with a Register request. See IEC 61334-4-511:2000, 7.2.

The Initiator system is identified with its System Title, MAC address and L-SAP selector:

initiator-descriptor::= structure
{
system_title: octet-string,
MAC_address: long-unsigned,
L-SAP_selector: unsigned
}

The size and the structure of the system title may be specified in system specifications. When the system title is used as part of the initialisation vector of cryptographic algorithms, then the size shall meet the requirements applicable for the initialisation vector.

The MAC_address element is used to update the *initiator-mac-address* MAC management variable when the system is configured in the synchronization-locked state. See the specification of the *initiator_mac_address* and the *synchronization_locked* attributes of the S-FSK Phy&MAC setup IC in 5.8.4.

As long as the server is not registered by an active initiator, the LSAP_selector field is set to 0 and the system_title field is equal to an octet string of 0s.

The default value of the initiator-descriptor is: system_title = octet-string of 0s, MAC_address = NO-BODY and L-SAP_selector = 0.

The value of this attribute can be updated by the invocation of the *reset_NEW_not_synchronized* method or by the CIASE Register service.

Method description

reset_NEW_not_synchronized (data)

Holds the MIB variable reset-NEW-not-synchronized (variable 17)

specified in IEC 61334-4-512:2001, 5.8.

Allows a client system to "reset" the server system. The submitted value corresponds to a client MAC address. The writing is refused if:

- the value does not correspond to a valid client MAC address or the predefined NO-BODY address;
- the submitted value is different from the NO-BODY address and the synchronization_locked attribute is not equal to TRUE.

For the description of the Intelligent Search Initiator process, see 10.7 of IEC 62056-8-3:—[9].

When this method is invoked, the following actions are performed:

- the system returns to the unconfigured state (UNC: MAC address equals NEW-address). This transition automatically causes the synchronization loss (function of the MAC sub-layer);
- the system changes the value of the active_initiator attribute: the MAC_address is set to the submitted value, the L-SAP_selector is set to value 0 and the system_title is set to an octet-string of 0s;
- all AAs that can be released are released.

5.8.6 S-FSK MAC synchronization timeouts (class_id: 52, version: 0)

An instance of the "S-FSK MAC synchronization timeouts" IC stores the timeouts related to the synchronization process.

S-FSK MAC synchronization timeouts	**0...n**	**class_id = 52, version = 0**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. search_initiator_timeout (static)	long-unsigned				x + 0x08
3. synchronization_confirmation_timeout (static)	long-unsigned				x + 0x10
4. time_out_not_addressed (static)	long-unsigned				x + 0x18
5. time_out_frame_not_OK (static)	long-unsigned				x + 0x20
Specific methods	***m/o***				

Attribute description

logical_name Identifies the "S-FSK synchronization timeouts" object instance. For logical names, see 6.2.20.

[9] To be published simultaneously with this part of IEC 62056.

search_initiator_ timeout	This timeout supports the intelligent search initiator function. It defines the value of the time, expressed in seconds, during which the server system is searching for the initiator with the strongest signal. During this timeout, all initiators, which may be heard by the servers, are expected to talk. After the expiry of this timeout, the server will accept a Register request from the initiator having provided the strongest signal and it will be locked to that initiator. If the value of the timeout is equal to 0, this means that the feature is not used. The timeout is started at the beginning of the Search Initiator Phase, when the server receives the first frame with a valid initiator MAC address. The timeout is restarted when the Search Initiator Phase is over and the server locks on the initiator. During the Check Initiator Phase, it is restarted on the reception of each valid frame. NOTE 1 A Fast synchronization can be performed if the level of signal is good enough (Level of initiator signal >= Search-Initiator-Threshold) and one of the MAC addresses (Source or Destination) is an Initiator MAC address. This means that the module (the meter) is next to a DC or next to a module that is already locked on that DC. The module locks in this case on that initiator.
synchronization_ confirmation_ timeout	Holds the MIB variable *synchronization-confirmation-timeout* (variable 6) specified in IEC 61334-4-512:2001, 5.3 and IEC 61334-5-1:2001, 4.3.7.6. Defines the value of the time, expressed in seconds, after which a server system which just gets frame synchronized (detection of a data path equal to AAAA54C7 hex) will automatically lose its frame synchronization if the MAC sublayer does not identify a valid MAC frame. The timeout starts after the reception of the first four bytes of a physical frame. The value of this variable can be modified by a client system. This timeout ensures a fast desynchronization of a system, which has synchronized on a wrong physical frame. See IEC 61334-5-1:2001, 3.5.3 for more details. NOTE 2 The default value of this variable can be specified in the implementation specifications. A value equal to 0 is equivalent to cancelling the use of the related synchronization_confirmation_timeout counter.

time_out_not_ addressed	Holds the MIB variable *time-out-not-addressed* (variable 7) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Defines the time, in minutes, after which a server system that has not been individually addressed: • returns to the non configured state (UNC: MAC-address equals NEW-address): this transition automatically involves the loss of the synchronization (function of the MAC sub-layer) and releasing all AAs that can be released; • loses its active initiator: the MAC address of the active-initiator is set to NO-BODY, the LSAP selector is set to the value 00 and the System Title is set to an octet-string of 0s. Because broadcast addresses are not individual system addresses, the timer associated with the *time-out-not-addressed* delay ensures that a forgotten system will sooner or later return to the unconfigured state. It will be then discovered again. A forgotten system is a system, which has not been individually addressed for more than the "*time-out-not-addressed*" amount of time. NOTE 3 The default value of this variable can be specified in the implementation specifications. A value equal to 0 is equivalent to cancelling the use of the related time-out-not-addressed counter.
time_out_frame_ not_OK	Holds the MIB variable *time-out-frame-not-OK* (variable 8), specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Defines the time, in seconds, after which a server system that has not received a properly formed MAC frame (incorrect NS field, inconsistent number of received sub-frames, false Cyclic Redundancy Code checking) loses its frame synchronization. The default value of this variable shall be specified in the implementation specifications. A value equal to 0 is equivalent to cancelling the use of the related *time-out-frame-not-OK* counter.

5.8.7 S-FSK MAC counters (class_id: 53, version: 0)

An instance of the “S-FSK MAC counters” IC stores counters related to the frame exchange, transmission and repetition phases.

S-FSK MAC counters		**0…n**	**class_id = 53, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. synchronization_register	(dyn.)	array				x + 0x08
3. desynchronization_listing	(dyn.)	structure				x + 0x10
4. broadcast_frames_counter	(dyn.)	array				x + 0x18
5. repetitions_counter	(dyn.)	double-long-unsigned			0	x + 0x20

S-FSK MAC counters	0...n	class_id = 53, version = 0			
6. transmissions_counter (dyn.)	double-long-unsigned			0	x + 0x28
7. CRC_OK_frames_counter (dyn.)	double-long-unsigned			0	x + 0x30
8. CRC_NOK_frames_counter (dyn.)	double-long-unsigned				x + 0x38
Specific methods	***m/o***				
1. reset (data)					x + 0x50

Attribute description

synchronization_ register

Holds the MIB variable *synchronization-register* (variable 23), specified in IEC 61334-4-512:2001, 5.8.

array synchronization-couples

synchronization-couples::= structure
{
    mac-address: long-unsigned,
    synchronizations-counter: double-long-unsigned
}

This variable counts the number of synchronization processes performed by the system. Processes that lead to a synchronization loss due to the detection of a wrong initiator are registered. The other processes that lead to a synchronization loss (time-out, management writing) **are not** registered.

This variable provides a balance sheet of the different systems on which the server system is "potentially" able to synchronize.

A synchronization process is initialized when the Management Application Entity (connection manager) receives a MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) primitive from the MAC Sublayer Entity. This process is registered in the synchronization-register variable only if the MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) primitive is followed by one of the three primitives:

1) MA_Data.indication (DA, SA, MSDU) primitive;
2) MA_Sync.indication (Synchronization State = SYNCHRO_CONF, SA, DA);
3) MA_Sync.indication (Synchronization State = SYNCHRO_LOSS, Synchro Loss Cause = wrong_initiator, SA, DA.

NOTE The third primitive is only generated if the server system is configured in a synchronization-locked state. See 5.8.4.

Processes which lead to the generation of MA_Sync.indication (Synchronization State = SYNCHRO_LOSS) primitives indicating synchronization loss due to:

- the physical layer;
- the time-out-not-addressed counter;
- setting the mac_address attribute of the S-FSK Phy&MAC setup object to NEW; see 5.8.4;
- or invoking the reset_NEW_not_synchronized method of the S-

FSK Active initiator object; see 5.8.5 (this is known as Management Writing)

are not taken into account in this variable.

For details on the MA_Sync.indication service primitive, see IEC 61334-5-1:2001, 4.1.7.1.

synchronization_ register (cont'd)

If the synchronization process ends with one of the three primitives listed above, the synchronization-register variable is updated by taking into account the SA and DA fields of the primitive.

The updating of the *synchronization-register* variable is carried out as follows:

First, the Management Entity checks the SA and DA fields.

- If one of these fields corresponds to a client MAC address (CMA) the Entity checks if the client MAC address (CMA) appears in one of the couples contained in the synchronization-register variable:
  - if it appears, the related synchronizations-counter subfield is incremented;
  - if it does not appear, a new (mac-address, synchronizations-counter) couple is added. This couple is initialized to the (CMA, 1) value.
- If none of the SA and DA fields correspond to a client MAC address, it is supposed that the system found its synchronization reference on a DiscoverReport type frame. In that case, the mac-address which should be registered in the synchronization-register variable is the predefined NEW value (0FFE). The updating of the synchronization-register variable is carried out in the same way as it is done for a normal client MAC address (CMA).

When a *synchronizations-counter* field reaches the maximum value, it automatically returns to 0 on the next increment.

The maximum number of synchronization couples {mac-address, synchronizations-counter} contained in this variable should be specified in the implementation specifications. When this maximum is reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the newest source MAC addresses are memorized.

The default value of this variable is an empty array.

desynchronization-listing

Holds the MIB variable *desynchronization-listing* (variable 24), specified in IEC 61334-4-512:2001, 5.8.

structure
{
nb_physical_layer_desynchronization: double_long_unsigned,
nb_time_out_not_addressed_desynchronization: double_long_unsigned,
nb_timeout_frame_not_OK desynchronization: double_long_unsigned,
nb_write_request_desynchronization: double_long_unsigned,
nb_wrong_initiator_desynchronization: double_long_unsigned
}

This variable counts the number of desynchronizations that occurred depending on their cause. On reception of synchronization loss

	notification, the Management Entity updates this attribute by incrementing the counter related to the cause of the desynchronization. When one of the counters reaches the maximum value, it automatically returns to 0 on the next increment. The default value of this variable contains elements, which are all equal to 0.

broadcast_frames_ counter	Holds the MIB variable *broadcast-frames-counter* (variable 19) specified in IEC 61334-4-512:2001, 5.8.
	
	array broadcast-couples
	
	broadcast-couples::= structure
	{
	source-mac-address: long-unsigned,
	frames-counter: double-long-unsigned
	}
	It counts the broadcast frames received by the server system and issued from a client system (source-mac-address = any valid client-mac-address, destination-mac address = ALL-physical). The number of frames is classified according to the origin of the transmitter. The counter is incremented even if the LLC-destination-address is not valid on the server system. When the frames-counter field reaches its maximum value, it automatically returns to 0 on the next increment. The maximum number of broadcast-couples {source-mac-address, frames-counter} contained in this variable should be specified in the implementation specifications. When this maximum is reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the newest source MAC addresses are memorized.

repetitions_ counter	Holds the MIB variable *repetitions-counter* (variable 20) specified in IEC 61334-4-512:2001, 5.8. Counts the number of repetition phases. The repetition phases following a transmission are not counted. If the MAC sub-layer is configured in the no-repeater mode, this variable is not updated. The repetitions counter measures the activity of the system as a repeater. A received frame repeated five times (from CC=4 to CC=0) is counted only once in the repetitions counter since it corresponds to one repetition phase. The counter is incremented at the beginning of each repetition phase. When the repetitions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
transmissions_ counter	Holds the MIB variable *transmissions-counter* (variable 21) specified in IEC 61334-4-512:2001, 5.8. Counts the number of transmission phases. A transmission phase is characterized by the transmission and the repetition of a frame. A repetition phase, which follows the reception of a frame is not counted. The transmission counter is incremented at the beginning of each transmission phase. A client system can write this variable to update the counter. When the transmissions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
CRC_OK_frames_ counter	Holds the MIB variable *CRC-OK-frames-counter* (variable 22) specified in IEC 61334-4-512:2001, 5.8

	Counts the number of frames received with a correct Frame Check Sequence Field. When the CRC OK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
CRC_NOK_frames_ counter	Counts the number of frames received with an incorrect Frame Check Sequence Field. When the CRC NOK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
Method description	
reset (data)	Clears all counters. data::= integer(0)

5.8.8 IEC 61334-4-32 LLC setup (class_id: 55, version: 1)

An instance of the "IEC 61334-4-32 LLC setup" IC holds parameters necessary to set up and manage the LLC layer as specified in IEC 61334-4-32.

IEC 61334-4-32 LLC setup	**0…n**	**class_id = 55, version = 1**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. max_frame_length (static)	long-unsigned				x + 0x08
3. reply_status_list (dyn.)	array				x + 0x10
Specific methods	***m/o***				

Attribute description

logical_name	Identifies the "IEC 61334-4-32 LLC setup" object instance. For logical names, see 6.2.20.
max_frame_length	Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4. For the S-FSK PLC profile, the minimum / default / maximum values are 26 bytes/ 134 bytes/ 242 bytes respectively. See IEC 61334-5-1:2001, 4.2.2. For other lower layer profiles, see the corresponding values in the relevant specification.

reply_status_list	Holds the MIB variable *reply-status-list* (variable 11) specified in IEC 61334-4-512:2001, 5.4.
	
	Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not already been read. The length of a waiting L-SDU is specified in number of sub-frames (different from zero). The variable is locally generated by the LLC sub-layer.
	
	array reply_status
	reply_status::= structure
	{
	L-SAP-selector: unsigned,
	length-of-waiting-L-SDU: unsigned
	}
	length-of-waiting-LSDU in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.

5.8.9 S-FSK Reporting system list (class_id: 56, version: 0)

An instance of the “S-FSK Reporting system list” IC holds the list of reporting systems.

S-FSK Reporting system list		**0...n**	**class_id = 56, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. reporting_system_list	(dyn.)	array				x + 0x08
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the “S-FSK Reporting system list” object instance. For logical names, see 6.2.20.

reporting_system_list	Holds the MIB variable *reporting-system-list* (variable 16) specified in IEC 61334-4-512:2001, 5.7. array system-title system-title::= octet-string Contains the system-titles of the server systems which have made a DiscoverReport request and which have not already been registered. The list has a finite size and it is sorted upon the arrival. The first element is the newest one. Once full, the oldest ones are replaced by the new ones. The reporting system list is updated: • when a DiscoverReport CI_PDU is received by the server system (whatever its state: non configured or configured): the CIASE adds the reporting system-title at the beginning of the list, and verifies that it does not exist anywhere else in the list, if so it destroys the old one. A system-title can only be present once in the list; • when a Register CI_PDU is received by the server system (whatever its state: non configured or configured): the CIASE checks the reporting-system list. If a system-title is present in the reporting-system-list and in the Register CI-PDU, the CIASE deletes the system-title in the reporting-system-list: this system is no more considered as a reporting system.

5.9 Interface classes for setting up the LLC layer for ISO/IEC 8802-2

5.9.1 General

This subclause 5.9 specifies the ICs available for setting up the ISO/IEC 8802-2 LLC layer, used in some DLMS/COSEM communication profiles, in the various types of operation.

5.9.2 Definitions related to the ISO/IEC 8802-2 LLC layer

See 1.4.2 of ISO/IEC 8802-2:1998.

5.9.3 ISO/IEC 8802-2 LLC Type 1 setup (class_id: 57, version: 0)

An instance of the ISO/IEC 8802-2 LLC Type 1 setup IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 1 operation.

ISO/IEC 8802-2 LLC Type 1 setup		**0…n**	**class_id = 57, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. max_octets_ui_pdu	(static)	long unsigned			128	x + 0x08
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the “ISO/IEC 8802-2 LLC Type 1 setup” object instance. For logical names, see 6.2.21.
max_octets_ui_pdu	Refer to the appropriate MAC protocol specification for any limitation on the maximum number of octets in a UI PDU. No restrictions are imposed by the LLC sublayer. However, in the interest of having a

value that all users of Type 1 LLC may depend upon, all MACs shall at least be capable of accommodating UI PDUs with information fields up to and including 128 octets in length.

See ISO/IEC 8802-2:1998, 6.8.1 *Maximum number of octets in a UI PDU.*

5.9.4 ISO/IEC 8802-2 LLC Type 2 setup (class_id: 58, version: 0)

An instance of the ISO/IEC 8802-2 LLC Type 2 setup IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 2 operation.

ISO/IEC 8802-2 LLC Type 2 setup		**0…n**	**class_id = 58, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. transmit_window_size_k	(static)	unsigned	1	127	1	x + 0x08
3. receive_window_size_rw	(static)	unsigned	1	127	1	x + 0x10
4. max_octets_i_pdu_n1	(static)	long unsigned			128	x + 0x18
5. max_number_transmissions_n2	(static)	unsigned				x + 0x20
6. acknowledgement_timer	(static)	long-unsigned				x + 0x28
7. p_bit_timer	(static)	long-unsigned				x + 0x30
8. reject_timer	(static)	long-unsigned				x + 0x38
9. busy_state_timer	(static)	long-unsigned				x + 0x40
Specific methods		***m/o***				

Attribute description

logical_name

Identifies the “ISO/IEC 8802-2 LLC Type 2 setup” object instance. For logical names, see 6.2.21.

transmit_window_size_k

The transmit window size (k) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of sequentially numbered I PDUs that the sending LLC may have outstanding (i.e., unacknowledged). The value of k is the maximum number by which the sending LLC send state variable V(S) can exceed the N(R) of the last received I PDU.

See ISO/IEC 8802-2:1998, 7.8.4 *Transmit window size, k.*

receive_window_size_rw

The receive window size (RW) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of unacknowledged sequentially numbered I PDUs that the local LLC allows the remote LLC to have outstanding. It is transmitted in the information field of XID (see ISO/IEC 8802-2:1998, 5.4.1.1.2) and applies to the XID sender. The XID receiver shall set its transmit window (k) to a value less than or equal to the receive window of the XID sender to avoid overrunning the XID sender.

See ISO/IEC 8802-2:1998, 7.8.6 *Receive window size, RW.*

max_octets_i_pdu_n1	N1 is a data link connection parameter that denotes the maximum number of octets in an I PDU. Refer to the various MAC descriptions to determine the precise value of N1 for a given medium access method. LLC itself places no restrictions on the value of N1. However, in the interest of having a value of N1 that all users of Type 2 LLC may depend upon, all MACs shall at least be capable of accommodating I PDUs with information fields up to an including 128 octets in length. See ISO/IEC 8802-2:1998, 7.8.3 *Maximum number of octets in an I PDU, N1.*
max_number_transmissions_n2	N2 is a data link connection parameter that indicates the maximum number of times that a PDU is sent following the running out of the acknowledgment timer, the P-bit timer, the reject timer, or the busy-state timer. See ISO/IEC 8802-2:1998, 7.8.2 *Maximum number of transmissions, N2.*
acknowledgement_timer	The acknowledgment timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive an acknowledgment to one or more outstanding I PDUs or an expected response PDU to a sent unnumbered command PDU. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.1 *Acknowledgement timer.*
p_bit_timer	The P-bit timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive a PDU with the F bit set to "1" in response to a sent Type 2 command with the P bit set to "1". The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.2 *P-bit timer.*
reject_timer	The reject timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive a reply to a sent REJ PDU (Reject Protocol Data Unit). The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.3 *Reject timer.*
busy_state_timer	The busy-state timer is a data link connection parameter that shall define the time interval during which the LLC shall wait for an indication of the clearance of a busy condition at the other LLC. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.4 *Busy-state timer.*

5.9.5 ISO/IEC 8802-2 LLC Type 3 setup (class_id: 59, version: 0)

An instance of the ISO/IEC 8802-2 LLC Type 3 setup IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 3 operation.

ISO/IEC 8802-2 LLC Type 3 setup		0...n	class_id = 59, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				X
2. max_octets_acn_pdu_n3	(static)	long unsigned				x + 0x08
3. max_number_transmissions_n4	(static)	unsigned				x + 0x10
4. acknowledgement_time_t1	(static)	long unsigned				x + 0x18
5. receive_lifetime_var_t2	(static)	long unsigned				x + 0x20
6. transmit_lifetime_var_t3	(static)	long unsigned				x + 0x28
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the "ISO/IEC 8802-2 LLC Type 3 setup" object instance. For logical names, see 6.2.21.
max_octets_acn_pdu_n3	N3 is a logical link parameter that denotes the maximum number of octets in an ACn command PDU. Refer to the various MAC descriptions to determine the precise value of N3 for a given medium access method. LLC places no restrictions on the value of N3. See ISO/IEC 8802-2:1998, 8.6.2 *Maximum number of octets in an ACn command PDU, N3*.
max_number_transmissions_n4	N4 is a logical link parameter that indicates the maximum number of times that an ACn command PDU is sent by LLC trying to accomplish a successful information exchange. Normally, N4 is set large enough to overcome the loss of a PDU due to link error conditions. If the medium access control sublayer has its own retransmission capability, the value of N4 may be set to 1 so that LLC does not itself requeue a PDU to the medium access control sublayer. See ISO/IEC 8802-2:1998, 8.6.1 *Maximum number of transmissions, N4*.

acknowledgement_ time_t1	The acknowledgment time is a logical link parameter that determines the period of the acknowledgment timers, and as such, shall define the time interval during which the LLC shall expect to receive an ACn response PDU from a specific LLC from which the LLC is awaiting a response PDU. The acknowledgment time shall take into account any delay introduced by the MAC sublayer and whether the timer is started at the beginning or at the end of the sending of the ACn command PDU by the LLC. The proper operation of the procedure shall require that the acknowledgment time be greater than the normal time between the sending of an ACn command PDU and the reception of the corresponding ACn response PDU. If the medium access control sublayer performs its own retransmissions and if the logical link parameter N4 is set to 1 to prevent LLC from re-queuing a PDU, then the acknowledgment time T1 may be set to infinity, making the acknowledgment timers unnecessary. The unit is seconds. Infinity is indicated by all bits set to 1. See ISO/IEC 8802-2:1998, 8.6.4 *Acknowledgement time, T1.*
receive_lifetime_ var_t2	This time value is a logical link parameter that determines the period of all of the receive variable lifetime timers. T2 shall be longer by a margin of safety than the longest possible period during which the first transmission and all retries of a single PDU may occur. The margin of safety shall take into account anything affecting LLCs perception of the arrival time of PDUs, such as LLC response time, timer resolution, and variations in the time required for the medium access control sublayer to pass received PDUs to LLC. If the destruction of the received state variables is not desired, the value of time T2 may be set to infinity. In this case the receive variable lifetime timer need not be implemented. The unit is seconds. Infinity is indicated by all bits set to 1. See ISO/IEC 8802-2:1998, 8.6.5 *Receive lifetime variable, T2.*

transmit_lifetime_ var_t3	This time value is a logical link parameter that determines the minimum lifetime of the transmit sequence state variables. T3 shall be longer by a margin of safety than: a) the logical link variable T2 at stations to which ACn commands are sent; and b) the longest possible lifetime of an ACn command-response pair. The lifetime of an ACn command-response pair shall take into account the sum of processing time, queuing delays, and transmission time for the command and response PDUs at the local and remote stations. If the destruction of the transmit state variables is not desired, the value of time T3 may be set to infinity. Note, if the receive variable lifetime parameter T2 is set to infinity at remote stations to which ACn commands are sent, then the T3 parameter shall be set to infinity at the local station. The unit is seconds. Infinity is indicated by all bits set to 1. See ISO/IEC 8802-2:1998, 8.6.6 *Transmit lifetime variables, T3.*

5.10 Maintenance of the interface classes

5.10.1 New versions of interface classes

Any modification of an existing IC affecting the transmission of service requests or responses results in a new version (version::= version+1) and shall be documented accordingly. The following rules shall be followed:

c) new attributes and methods may be added;

d) 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;

e) if these rules cannot be met, then a new IC shall be created;

Any modification of ICs will be recorded by moving the old version of an IC into Clause 7.

5.10.2 New interface classes

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

5.10.3 Removal of interface classes

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

6 Relation to OBIS

6.1 General

This Clause 6 specifies the use of COSEM interface objects to model various data items.

It also specifies the logical names of the objects. The naming system is based on OBIS, the Object Identification System: each logical name is an OBIS code.

OBIS codes are specified in the following subclauses:

- 6.2 specifies the use and the logical names of abstract COSEM objects, i.e. objects not related to an energy type;
- 6.3 specifies the use and logical names for electricity related COSEM objects;
- the detailed OBIS code allocations are specified in IEC 62056-6-1.

Unless otherwise specified, the use of value group B shall be:

- if just one object is instantiated, the value in value group B shall be 0;
- if more than one object is instantiated in the same physical device, the value group B shall number the measurement or communication channels as appropriate, from 1 to 64. This is indicated by the letter “b”.

Unless otherwise specified the use of value group E shall be:

- if just one object is instantiated, value in value group E shall be 0;
- if more than one object is instantiated in the same physical device, the value group E shall number the instantiations from zero to the maximum value needed. This is indicated by the letter “e”. For the values allocated, see IEC 62056-6-1.

All codes, which are not explicitly listed, but which are outside the manufacturer, utility or consortia specific ranges are reserved for future use.

6.2 Abstract COSEM objects

6.2.1 Use of value group C

Table 12 shows the use of value group C for abstract objects in the COSEM context. See also IEC 62056-6-1:—, Table 4.

Table 12 – Use of value group C for abstract objects in the COSEM context

Value group C Abstract objects (A = 0)	
0	General purpose COSEM objects
1	Instances of IC "Clock"
2	Instances of IC “Modem configuration” and related IC-s
10	Instances of IC "Script table"
11	Instances of IC "Special days table"
12	Instances of IC "Schedule"
13	Instances of IC "Activity calendar"
14	Instances of IC “Register activation”
15	Instances of IC "Single action schedule"
16	Instances of IC “Register monitor”
17	Instances of IC “Limiter”
20	Instances of IC "IEC local port setup"
21	Standard readout definitions
22	Instances of IC “IEC HDLC setup”
23	Instances of IC "IEC twisted pair (1) setup"
24	COSEM objects related to M-Bus
25	Instances of IC “TCP-UDP setup”, “IPv4 setup”, “MAC address setup”, “PPP setup”, “GPRS modem setup”, “SMTP setup”
26	COSEM objects for S-FSK PLC setup
27	COSEM objects for ISO/IEC 8802-2 LLC layer setup
31	Instances of IC “Wireless Mode Q”
40	Instances of IC “Association SN/LN”
41	Instances of IC “SAP assignment”
42	COSEM logical device name
43	Instances of IC “Security setup”
44	Instances of IC “Image transfer”
65	Instances of IC “Utility tables”
67	Instances of “Sensor manager”
128…199	Manufacturer specific COSEM related abstract objects
All other	Reserved

6.2.2 Data of historical billing periods

COSEM provides three mechanisms to represent values of historical billing periods. This is shown in Figure 16 and is described below:

- a value of a single historical billing period may be represented using the same IC as used for representing the value of the current billing period. With F = 0…99, the billing period is identified by the value of the billing period counter VZ. F = VZ identifies the youngest value, F = VZ-1 identifies the second youngest value, etc. F = 101…125 identifies the last, second last, …25th last billing period. (F = 255 identifies the current billing period). Simple

objects can only be used to represent values of historical billing periods, if “Profile generic” objects are not implemented;

- a value of a single historical billing period may also be represented by “Profile generic” objects, which are one entry deep, and contain the historical value itself and the time stamp of the storage. With F = 0…99, the billing period is identified by the value of the billing period counter VZ. F = VZ identifies the youngest value, F = VZ-1 identifies the second youngest value, etc. F=101 identifies the most recent billing period;
- values of multiple historical billing periods are represented with “Profile generic” objects, with suitable controlling attributes. With F = 102…125 the two last, …25 last values can be reached. F = 126 identifies an unspecified number of historical values;
- when values of historical billing periods are represented by “Profile generic” objects, more than one billing periods schemes may be used. The billing period scheme is identified by the billing period counter object captured in the profile.

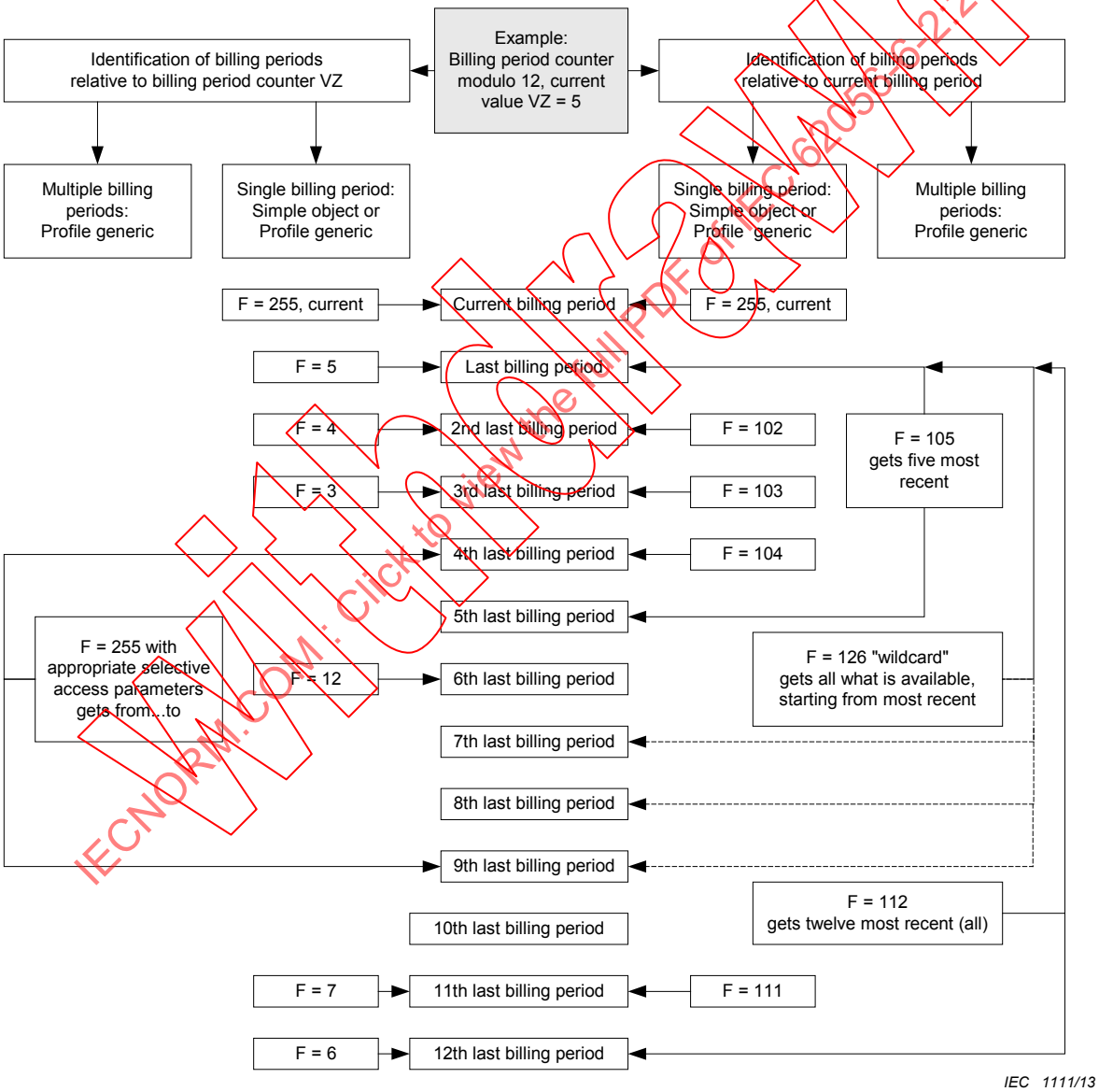


Figure 16 – Data of historical billing periods – example with module 12, VZ = 5

6.2.3 Billing period values / reset counter entries

These values are represented by instances of the IC "Data” with data type *unsigned* respectively *octet-string*, formatted as *date_time,* see 4.6.1. They may be also related to an energy type. See also 6.3.3. When the values of historical periods are represented by “Profile

generic" objects, the time stamp of the billing period objects shall be part of the captured objects. The values can also be related to a channel.

Billing period values / reset counter entries	IC	OBIS code					
		A	B	C	D	E	F
For item names and OBIS codes, see IEC 62056-6-1:—, Table 7.	1, Data [a]	0	*b*	0	1	*e*	255

[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.4 Clock objects (class_id: 8)

Instances of the IC "Clock" – see 5.4.1 – control the system clock of the physical device.

Clock objects	IC	OBIS code					
		A	B	C	D	E	F
Clock	8, Clock	0	*b*	1	0	*e*	255

6.2.5 Modem configuration and related objects

Instances of the IC "Modem configuration" – see 5.5.4 – define and control the behaviour of the device regarding the communication through a modem.

Instances of the IC "Auto connect" – see 5.5.6 – define the necessary parameters for the management of sending information from the metering device to one or more destinations.

Instances of the IC "Auto answer" – see 5.5.5 – define and control the behaviour of the device regarding the auto answering function using a modem.

Modem configuration and related objects	IC	OBIS code					
		A	B	C	D	E	F
Modem configuration	27, Modem configuration	0	*b*	2	0	0	255
Auto connect	29, Auto connect	0	*b*	2	1	0	255
Auto answer	28, Auto answer	0	*b*	2	2	0	255

6.2.6 Script table objects (class_id: 9)

Instances of the IC "Script table" – see 5.4.2 – control the behaviour of the device.

Several instances are predefined and normally available as hidden scripts only with access to the execute() method. The table below contains only the identifiers for the "standard" instances of the listed scripts. Implementation specific instances of these scripts should use values different from zero in value group D.

The *MDI reset / End of billing period* script table defines the actions to be performed at the end of the billing period, for example the reset of maximum demand indicator registers and archiving data.

If there are several billing period schemes available, then there shall be one script present in the array of scripts for each billing period scheme.

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

The *Disconnect control* script table holds the scripts to invoke the methods of Disconnect control objects.

The *Image activation* script table is used to locally activate an Image transferred to the client, at the date and time held by an Image activation "Single action schedule" object.

The *Broadcast* script table allows standardizing utility wide the entry point into regularly needed functionality.

Script table objects	IC	OBIS code					
		A	B	C	D	E	F
Global meter reset [a]	9, Script table	0	*b*	10	0	0	255
MDI reset / End of billing period [a]		0	*b*	10	0	1	255
Tariffication script table		0	*b*	10	0	100	255
Activate test mode [a]		0	*b*	10	0	101	255
Activate normal mode [a]		0	*b*	10	0	102	255
Set output signals		0	*b*	10	0	103	255
Switch optical test output [b, c]		0	*b*	10	0	104	255
Power quality measurement management		0	*b*	10	0	105	255
Disconnect control		0	*b*	10	0	106	255
Image activation		0	*b*	10	0	107	255
Broadcast script table		0	*b*	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.

[b] The optical test output is switched to measuring quantity Y and the test mode is activated by calling the execute method of the script table object 0.x.10.0.104.255 using Y as parameter; where Y is given by IEC 62056-6-1:—, 7.1, Table 12. The default value of A is 1 (Electricity).

EXAMPLE In the case of electricity meters, A = 1, default, execute (21) switches the test output to display the active power + of phase 1.

[c] The optical test output is also switched back to its default value when this script is activated.

6.2.7 Special days table objects (class_id: 11)

Instances of the IC "Special days table" – see 5.4.4 – define and control the behaviour of the device regarding calendar functions on special days for clock control.

Special days table objects	IC	OBIS code					
		A	B	C	D	E	F
Special days table	11, Special days table	0	*b*	11	0	0	255

6.2.8 Schedule objects (class_id: 10)

Instances of the IC "Schedule" – see 5.4.3 – define and control the behaviour of the device in a sequenced way.

Schedule objects	IC	OBIS code					
		A	B	C	D	E	F
Schedule	10, Schedule	0	*b*	12	0	*e*	255

6.2.9 Activity calendar objects (class_id: 20)

Instances of the IC "Activity calendar" – see 5.4.5 – define and control the behaviour of the device in a calendar-based way.

Activity calendar objects	IC	OBIS code					
		A	B	C	D	E	F
Activity calendar	20, Activity calendar	0	*b*	13	0	0	255

6.2.10 Register activation objects (class_id: 6)

Instances of the IC "Register activation"– see 5.2.5 – are used to handle different tariffication structures.

Register activation objects	IC	OBIS code					
		A	B	C	D	E	F
Register activation	6, Register activation	0	*b*	14	0	*e*	255

6.2.11 Single action schedule objects (class_id: 22)

Instances of the IC "Single action schedule" – see 5.4.7 – control the behaviour of the device. Implementation specific instances should use values different from zero in value group D.

Single action schedule objects	IC	OBIS code					
		A	B	C	D	E	F
End of billing period	22, Single action schedule	0	*b*	15	0	0	255
Disconnect control scheduler		0	*b*	15	0	1	255
Image activation		0	*b*	15	0	2	255
Output control scheduler		0	*b*	15	0	3	255

6.2.12 Register monitor objects (class_id: 21)

Instances of the IC "Register monitor" – see 5.4.6 – control the register monitoring function of the device. They define the value to be monitored, the set of thresholds to which the value is compared, and the actions to be performed when a threshold is crossed.

In general, the logical name(s) shown in the table below shall be used. See also 6.3.9 and 6.3.10.

Register monitor objects	IC	OBIS code					
		A	B	C	D	E	F
Register monitor	21, Register monitor	0	*b*	16	0	*e*	255

6.2.13 Limiter objects (class_id: 71)

Instances of the IC "Limiter" handle the monitoring of values in normal and emergency conditions. See also 5.4.9.

Limiter objects	IC	OBIS code					
		A	B	C	D	E	F
Limiter	71, Limiter	0	*b*	17	0	*e*	255

6.2.14 IEC local port setup objects (class_id: 19)

These objects define and control the behaviour of local ports using the protocol specified in IEC 62056-21. See also 5.5.1.

IEC local port setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC optical port setup	19, IEC local port setup	0	*b*	20	0	0	255
IEC electrical port setup		0	*b*	20	0	1	255

6.2.15 Standard readout profile objects (class_id: 7)

A set of objects is defined to carry the standard readout as it would appear with IEC 62056-21:2002 (modes A to D). See also 5.2.6.

Standard readout objects	IC	OBIS code					
		A	B	C	D	E	F
General local port readout	7, Profile generic	0	*b*	21	0	0	255
General display readout		0	*b*	21	0	1	255
Alternate display readout		0	*b*	21	0	2	255
Service display readout		0	*b*	21	0	3	255
List of configurable meter data		0	*b*	21	0	4	255
Additional readout profile 1		0	*b*	21	0	5	255
.....							
Additional readout profile *n*		0	*b*	21	0	N	255

For the parametrization of the standard readout, “Data” objects can be used.

Standard readout parametrization objects	IC	OBIS code					
		A	B	C	D	E	F
Standard readout parametrization	1, Data	0	*b*	21	0	*e*	255

6.2.16 IEC HDLC setup objects (class_id: 23)

Instances of the IC “IEC HDLC setup” – see 5.5.2 – hold the parameters of the HLDC based data link layer.

IEC HDLC setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC HDLC setup	23, IEC HDLC setup	0	*b*	22	0	0	255

6.2.17 IEC twisted pair (1) setup objects (class_id: 24)

Instances of the IC “IEC twisted pair (1) setup” – see 5.5.3 – define and control the behaviour of local ports using the protocol specified in Draft IEC 62056-3-1.

IEC twisted pair (1) setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC twisted pair (1) setup	24, IEC twisted pair (1) setup	0	*b*	23	0	0	255

6.2.18 Objects related to data exchange over M-Bus

The following objects are available to model and control data exchange using the M-Bus protocol specified in EN 13757-3 and EN 13757-5.

- instances of the IC “M-Bus slave port setup” define and control the behaviour of M-Bus slave ports of a DLMS/COSEM device. See 5.6.1;
- instances of the IC “M-Bus client” are used to configure DLMS/COSEM devices as M-Bus clients. There is one M-Bus client object for each M-Bus slave. Value group B identifies the M-Bus channels. See 5.6.2;
- “M-Bus value” objects, instances of the IC “Extended register”, hold the values captured from M-Bus slave devices on the relevant channel. The link between the M-Bus client setup objects and the M-Bus value objects is provided by the channel number.
- “M-Bus profile generic” objects capture M-Bus value objects possibly along with other, not M-Bus specific objects;
- “M-Bus disconnect control” objects control disconnect devices of M-Bus devices (e.g. gas valves);
- instances of the IC “Wireless mode Q” define and control the behaviour of the device regarding the communication parameters according to mode Q of EN 13757-5. A node

having more than one network address, i.e. a multi-homed node, will have multiple objects of this types. See 5.6.3;

- "M-Bus control log" objects log the changes of the state of the disconnect devices;
- instances of the IC "M-Bus master port setup" define and control the behaviour of M-Bus master ports of DLMS/COSEM devices, allowing to exchange data with M-Bus slaves. See 5.6.4.

Objects related to data exchange over M-Bus	IC	OBIS code					
		A	B	C	D	E	F
M-Bus slave port setup	25, M-Bus slave port setup	0	*b*	24	0	0	255
M-Bus client	72, M-Bus client	0	*b*	24	1	0	255
M-Bus value	4, Extended register	0	*b*	24	2	e [a]	255
M-Bus profile generic	7, Profile generic	0	*b*	24	3	e	255
M-Bus disconnect control	70, Disconnect control	0	*b*	24	4	0	255
M-Bus control log	7, Profile generic	0	*b*	24	5	0	255
M-Bus master port setup	74, M-Bus master port setup	0	*b*	24	6	0	255
Wireless Mode Q channel	73, Wireless Mode Q channel	0	*b*	31	0	0	255

[a] "e" is equal to the index of the captured value in accordance to index of capture_definition_element in the capture_definition attribute of the M-Bus client object.

6.2.19 Objects to set up data exchange over the Internet

Instances of the IC "TCP-UDP setup" – see 5.7.1 – handle all information related to the setup of the TCP and UDP layer of the Internet based communication profile(s), and point to the IP setup object(s) handling the setup of the IP layer on which the TCP-UDP connection(s) is (are) used.

Instances of the IC "IPv4 setup" – see 5.7.2 – handle all information related to the setup of the IPv4 layer of the Internet based communication profile(s) and point to the data link layer setup object(s) handling the setup of the data link layer on which the IP connections are used.

Instances of the IC "MAC address setup" – see 5.7.3 – handle all information related to the setup of the Ethernet data link layer of the Internet based communication profile(s).

Instances of the IC "PPP setup" – see 5.7.4 – handle all information related to the setup of the PPP data link layer of the Internet based communication profiles.

Instances of the IC "GPRS modem setup" – see 5.7.5 – handle all information related to the setup of the GPRS modem.

Instances of the IC "SMTP setup" – see 5.7.6 – handle all information related to the setup of the SMTP service.

Objects to set up data exchange over the Internet	IC	OBIS code					
		A	B	C	D	E	F
TCP-UDP setup	41, TCP-UDP setup	0	*b*	25	0	0	255
IPv4 setup	42, IPv4 setup	0	*b*	25	1	0	255
MAC address setup	43, MAC address setup	0	*b*	25	2	0	255
PPP setup	44, PPP setup	0	*b*	25	3	0	255
GPRS modem setup	45, GPRS modem setup	0	*b*	25	4	0	255
SMTP setup	46, SMTP setup	0	*b*	25	5	0	255

6.2.20 Objects for setting up data exchange using S-FSK PLC

Instances of the IC "S-FSK Phy&MAC setup" – see 5.8.4 – handle all information related to setting up the PLC S-FSK lower layer profile specified in IEC 61334-5-1.

Instances of the IC "S-FSK Active initiator" – see 5.8.5 – handle all information related to the active initiator in the PLC S-FSK lower layer profile specified in IEC 61334-5-1.

Instances of the IC "S-FSK MAC synchronization timeouts" – see 5.8.6 – manage all timeouts related to the synchronization process of devices using the PLC S-FSK lower layer profile specified in IEC 61334-5-1.

Instances of the IC "S-FSK MAC counters" – see 5.8.7 – store counters related to the frame exchange, transmission and repetition phases in the PLC S-FSK lower layer profile specified in IEC 61334-5-1.

Instances of the IC "IEC 61334-4-32 LLC setup" – see 5.8.8 – handle all information related to the LLC layer specified in IEC 61334-4-32.

Instances of the IC "S-FSK Reporting system list" – see 5.8.9 – hold information on reporting systems in the PLC S-FSK lower layer profile specified in IEC 61334-5-1.

Objects to set up data exchange using S-FSK PLC	IC	OBIS identification					
		A	B	C	D	E	F
S-FSK Phy&MAC setup	50, S-FSK Phy&MAC setup	0	*b*	26	0	0	255
S-FSK Active initiator	51, S-FSK Active initiator	0	*b*	26	1	0	255
S-FSK MAC synchronization timeouts	52, S-FSK MAC synchronization	0	*b*	26	2	0	255
S-FSK MAC counters	53, S-FSK MAC counters	0	*b*	26	3	0	255
NOTE Placeholder for a Monitoring IC to be specified.							
IEC 61334-4-32 LLC setup	55, IEC 61334-4-32 LLC setup	0	*b*	26	5	0	255
S-FSK Reporting system list	56, S-FSK Reporting system list	0	*b*	26	6	0	255

6.2.21 Objects for setting up the ISO/IEC 8802-2 LLC layer

Instances of the IC "ISO/IEC 8802-2 LLC Type 1 setup" – see 5.9.3 – handle all information related to the LLC layer specified in ISO/IEC 8802-2 in Type 1 operation.

Instances of the IC "ISO/IEC 8802-2 LLC Type 2 setup" – see 5.9.4 – handle all information related to the LLC layer specified in ISO/IEC 8802-2 in Type 2 operation.

Instances of the IC "ISO/IEC 8802-2 LLC Type 3 setup" – see 5.9.5 – handle all information related to the LLC layer specified in ISO/IEC 8802-2 in Type 3 operation.

Objects to set up the ISO/IEC 8802-2 LLC layer	IC	OBIS identification					
		A	B	C	D	E	F
ISO/IEC 8802-2 LLC Type 1 setup	57, ISO/IEC 8802-2 LLC Type 1 setup	0	*b*	27	0	0	255
ISO/IEC 8802-2 LLC Type 2 setup	58, ISO/IEC 8802-2 LLC Type 2 setup	0	*b*	27	1	0	255
ISO/IEC 8802-2 LLC Type 3 setup	59, ISO/IEC 8802-2 LLC Type 3 setup	0	*b*	27	2	0	255

6.2.22 Association objects (class_id: 12, 15)

A series of Association SN / LN objects – see 5.3.1, 5.3.2 – is available to model application associations between a COSEM client and server.

Association objects	IC	OBIS code					
		A	B	C	D	E	F
Current association	12, Association SN 15, Association LN	0	0	40	0	0	255
Association, instance 1		0	0	40	0	1	255
……							
Association, instance *n*		0	0	40	0	n	255

6.2.23 SAP assignment object (class_id: 17)

An instance of the IC "SAP assignment" – see 5.3.3 – holds information about the addresses (Service Access Points, SAPs) of logical devices within a physical device.

SAP Assignment object	IC	OBIS code					
		A	B	C	D	E	F
SAP assignment of current physical device	17, SAP assignment	0	0	41	0	0	255

6.2.24 COSEM logical device name object

Each COSEM logical device shall be identified by its logical device name, unique worldwide. See 4.8.2. It is held by the *value* attribute of a "Data" or "Register" object, with data type *octet-string*. For short name referencing, the base_name of the object is fixed. See 4.3.

COSEM logical device name object	IC	OBIS code					
		A	B	C	D	E	F
COSEM logical device name	1, Data [a]	0	0	42	0	0	255
[a] If the IC "Data" is not available, the IC "Register" (with scaler = 0, unit = 255) may be used.							

6.2.25 Security setup and frame counter objects (class_id: 64)

Instances of the IC "Security setup" – see 5.3.5 – are used to set up the message security features. For each Association object, there is one Security setup object managing security within that AA. See 7.4 and 7.5. Value group E numbers the instances.

Security setup objects	IC	OBIS identification					
		A	B	C	D	E	F
Security setup	64, Security setup	0	0	43	0	*e*	255

Frame counter objects hold the frame counter element of the initialization vector. They are instances of the IC "Data". The value in value group B identifies the communication channel.

NOTE The same client can use different communication channels e.g. a remote port and a local port. The frame counter on the different channels can be different.

The value in value group E shall be the same as in the logical name of the corresponding Security setup object.

Frame counter objects	IC	OBIS code					
		A	B	C	D	E	F
Fame counter	1, Data [a]	0	*b*	43	1	*e*	255
[a] If the IC "Data" is not available, the IC "Register" (with scaler = 0, unit = 255) may be used.							

6.2.26 Image transfer objects (class_id: 18)

Instances of the IC "Image transfer" – see 5.3.4 – control the Image transfer process.

Image transfer related objects	IC	OBIS identification					
		A	B	C	D	E	F
Image transfer	18, Image transfer	0	0	44	0	*e*	255

6.2.27 Utility table objects (class_id: 26)

Instances of the IC "Utility tables" – see 5.2.7 – allow to model ANSI utility tables. The Utility table IDs are mapped to OBIS codes as follows:

- value group A: use value of 0 to specify abstract object;
- value group B: instance of table set;
- value group C: use value 65 – signifies utility tables specific definitions;
- value group D: table group selector;
- value group E: table number within group;
- value group F: use value 0xFF for data of current billing period.

Utility table objects	IC	OBIS code					
		A	B	C	D	E	F
Standard tables 0-127	26, Utility tables	0	*b*	65	0	*e*	255
Standard tables 128-255		0	*b*	65	1	*e*	255
…							
Standard tables 1920-2047		0	*b*	65	15	*e*	255
Manufacturer tables 0-127		0	*b*	65	16	*e*	255
Manufacturer tables 128-255		0	*b*	65	17	*e*	255
…							
Manufacturer tables 1920-2047		0	*b*	65	31	*e*	255
Standard pending tables 0-127		0	*b*	65	32	*e*	255
Standard pending tables 128-255		0	*b*	65	33	*e*	255
…							
Standard pending tables 1920-2047		0	*b*	65	47	*e*	255
Manufacturing pending tables 0-127		0	*b*	65	48	*e*	255
Manufacturing pending tables 128-255		0	*b*	65	49	*e*	255
…							
Manufacturing pending tables 1920-2047		0	*b*	65	63	*e*	255

6.2.28 Device ID objects

A series of objects are used to hold ID numbers of the device. These ID numbers can be defined by the manufacturer (e.g. manufacturing number) or by the user.

They are held by the *value* attribute of "Data" objects, with data type *octet-string*. If more than one of those is used, it is allowed to combine them into a "Profile generic" object. In this case, the captured objects are *value* attributes of the device ID “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, a “Register table” object – see 5.2.8 – can be used. See also in IEC 62056-6-1:—, Table 7.

Device ID objects	IC	OBIS code					
		A	B	C	D	E	F
Device ID 1…10 object (manufacturing number)	1, Data [a]	0	*b*	96	1	0…9	255
Device ID-s object	7, Profile generic	0	*b*	96	1	255	255
Device ID-s object	61, Register table	0	*b*	96	1	255	255
[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

6.2.29 Metering point ID objects

One object is available to store a media type independent metering point ID. It is held by the *value* attribute of a “Data” object, with data type *octet-string*.

Metering point ID objects	IC	OBIS code					
		A	B	C	D	E	F
Metering point ID	1, Data [a]	0	*b*	96	1	10	255
[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

6.2.30 Parameter changes and calibration objects

A set of simple COSEM objects describes the history of the configuration of the device. All values are modelled by instances of the IC "Data".

Parameter changes objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7	1, Data [a]	0	*b*	96	2	*e*	255
[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

6.2.31 I/O control signal objects

A series of objects are available to define and control the status of I/O lines of the physical metering equipment.

The status is held by the *value* attribute of a “Data” object, with data type *octet-string* or *bit-string*. Alternatively, the status is held by a “Status mapping” object, see 5.2.9, which holds both the status word and the mapping of its bits to the reference table. If there are several I/O control status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global state of I/O control signals object. See also IEC 62056-6-1:—, Table 7.

I/O control signal objects	IC	OBIS code					
		A	B	C	D	E	F
I/O control signal objects, contents manufacturer specific	1, Data [a]	0	*b*	96	3	0…4	255
I/O control signal objects, contents mapped to a reference table	63, Status mapping	0	*b*	96	3	0…4	255
I/O control signal objects, global	7, Profile generic or 61, Register table	0	*b*	96	3	0	255

[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.32 Disconnect control objects (class_id: 70)

Instances of the IC "Disconnect control" – see 5.4.8 – manage internal or external disconnect units (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely – the premises of the consumer to / from the supply. See also 6.2.18.

Disconnect control objects	IC	OBIS code					
		A	B	C	D	E	F
Disconnect control	70, Disconnect control	0	b	96	3	10	255

6.2.33 Status of internal control signals objects

A series of objects are available to hold the status of internal control signals.

The status carries binary information from a bitmap, and it shall be held by the *value* attribute of a "Data" object, with data type *bit-string, unsigned, long-unsigned, double-long-unsigned, long64-unsigned or octet-string.* Alternatively, the status is held by a "Status mapping" object, see 5.2.9, which holds both the status word and the mapping of its bits to the reference table. If there are several status of internal control signals objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global status of the internal control signals object. See also IEC 62056-6-1:—, Table 7.

Internal control signal objects	IC	OBIS code					
		A	B	C	D	E	F
Internal control signals, contents manufacturer specific	1, Data [a]	0	*b*	96	4	0…4	255
Internal control signals, contents mapped to a reference table	63, Status mapping	0	*b*	96	4	0…4	255
Internal control signals, global	7, Profile generic or 61, Register table	0	*b*	96	4	0	255

[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.34 Internal operating status objects

A series of objects are available to hold internal operating statuses.

The status carries binary information from a bitmap, and it shall be held by the value attribute of a "Data" object, with data type *bit-string, unsigned, long-unsigned, double-long-unsigned, long64-unsigned or octet-string.* Alternatively, the status is held by a "Status mapping" object, see 5.2.9, which holds both the status word and the mapping of its bits to the reference table. If there are several internal operating status objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global internal operating status object. See also IEC 62056-6-1:—, Table 7.

Internal operating status objects	IC	OBIS code					
		A	B	C	D	E	F
Internal operating status objects, contents manufacturer specific	1, Data [a]	0	*b*	96	5	0…4	255
Internal operating status objects, contents mapped to a reference table	63, Status mapping	0	*b*	96	5	0…4	255
Internal operating status objects, global	7, Profile generic or 61, Register table	0	*b*	96	5	0	255
[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.							

Internal operating status objects can also be related to an energy type. See 6.3.7.

6.2.35 Battery entries objects

A series of objects are available for holding information related to the battery of the device. These objects are instances of IC "Data", "Register" or "Extended register" as appropriate.

Battery entries objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register or 4, Extended register	0	*b*	96	6	0…6	255

6.2.36 Power failure monitoring objects

A series of objects are available for power failure monitoring.

For simple power failure monitoring, it is possible to count the number of power failure events affecting all three phases, one of the three phases, any of the phases, and the auxiliary supply.

For advanced power failure monitoring, it is possible to define a time threshold to make a distinction between short and long power failure events. It is possible to count the number of such long power failure events separately from the short ones, as well as to store their time of occurrence and duration (time from power down to power up) in all three phases, in one of the three phases and in any of the phases.

The number of power failure events objects are represented by instances of the IC "Data", "Register" or "Extended register" with data types *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

The power failure duration, time and time threshold data are represented by instances of the IC "Data", "Register" or "Extended register" with appropriate data types.

If power failure duration objects are represented by instances of the IC "Data", then the default scaler shall be 0, and the default unit shall be the *second*.

Power failure monitoring objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register or 4, Extended register	0	*b*	96	7	0… 21	255

These objects may be collected in a "Power failure event log" object. See IEC 62056-6-1:—, Table 22.

6.2.37 Operating time objects

A series of objects are available for holding the cumulated operating time and the various tariff registers of the device. These objects are instances of the IC “Data”, ”Register” or “Extended register”. The data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. If the IC “Data” is used, the unit shall be the second by default.

Operating time objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register or 4, Extended register	0	*b*	96	8	0… 63	255

6.2.38 Environment related parameters objects

A series of objects are available to store environmental related parameters. They are held by the *value* attribute of instances of the IC “Register” or “Extended register”, with appropriate data types.

Environment related parameters objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	3, Register or 4, Extended register	0	*b*	96	9	0…2	255

6.2.39 Status register objects

A series of objects are available to hold statuses that can be captured in load profiles. See also IEC 62056-6-1:—, Table 7.

Status register objects	IC	OBIS code					
		A	B	C	D	E	F
Status register, contents manufacturer specific	1, Data [a]	0	*b*	96	10	1… 10	255
Status register, contents mapped to reference table	63, Status mapping	0	*b*	96	10	1… 10	255

[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

The status register is held by the value attribute of a “Data” object, with data type *bit-string, unsigned, long-unsigned, double-long-unsigned, long64-unsigned* or *octet-string*. It carries binary information from a bitmap. Its contents is not specified.

Alternatively, the status register may be held by the *status_word* attribute of a “Status mapping” object, see 5.2.9.The *mapping_table* attribute holds mapping information between the bits of the status word and entries of a reference table.

6.2.40 Event code objects

In the meter or in its environment, various events may be generated.

A series of objects are available to hold an identifier of a most recent event (event code). Different instances of event code objects may be captured in different instances of event logs; see 6.2.48.

NOTE The definition of event identifiers is out of the Scope of this standard.

Event code objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register, or 4, Extended register	0	*b*	96	11	0…9	255

Events may also set flags in error registers and alarm registers. See also 6.2.46.

6.2.41 Communication port log parameter objects

A series of objects are available to hold various communication log parameters. They are represented by instances of IC “Data”, “Register” or “Extended register”.

Communication port log parameter objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register, or 4, Extended register	0	*b*	96	12	0…6	255

6.2.42 Consumer message objects

A series of objects are available to store information sent to the energy end-user. The information may appear on the display of the meter and / or on a consumer information port.

Consumer message objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register, or 4, Extended register	0	*b*	96	13	0, 1	255

6.2.43 Currently active tariff objects

A series of objects are available to hold the identifier of the currently active tariff. They carry the same information as the active_mask attribute of the corresponding Register activation object.

Currently active tariff objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register, or 4, Extended register	0	*b*	96	14	0… 15	255

6.2.44 Event counter objects

A series of objects are available to count events. The number of the events is held by the value attribute.

Event counter objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 7.	1, Data, 3, Register, or 4, Extended register	0	*b*	96	15	0…9	255

6.2.45 Error register objects

A series of objects are used to communicate error indications of the device.

The different error registers are held by the *value* attribute of “Data” objects, with data type *bit-string*, *octet-string, unsigned, long-unsigned, double-long-unsigned* or *long64-unsigned.*

The individual bits of the error register may be set and cleared by a pre-defined selection of events – see 6.2.40. Depending on the type of the error, some errors may clear themselves when the reason setting the error flag disappears.

If more than one of those objects is used, it is allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are the *value* attributes “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC “Register table” can be used.

Error register objects can also be related to an energy type and to a channel. See IEC 62056-6-1:—, 6.2 and 7.5.2.

Error register objects	IC	OBIS code					
		A	B	C	D	E	F
Error register 1…10 object	1, Data [a]	0	b	97	97	0…9	255
Error profile object	7, Profile generic	0	b	97	97	255	255
Error table object	61, Register table	0	b	97	97	255	255

[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.2.46 Alarm registers and alarm filters objects

A number of objects are available to hold alarm registers. The different alarm registers are held by the *value* attribute of “Data” objects, with data type *bit-string*, *octet-string, unsigned, long-unsigned, double-long-unsigned* or *long64-unsigned.* When selected events occur, they set the corresponding flag and the device raises an alarm. Alarm flags do not re-set themselves; they can be reset by writing the *value* attribute only.

If more than one of those objects is used, it is also allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are the “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC “Register table” can be used.

Alarm filter objects are available to define if an event is to be handled as an alarm when it appears. The different alarm filters are held by the *value* attribute of “Data” objects, with data type *bit-string, octet-string, unsigned, long-unsigned, double-long-unsigned or long64-unsigned.* The bit mask has the same structure as the corresponding alarm register object. If a bit in the alarm filter is set, then the corresponding alarm is enabled, otherwise it is disabled.

See also IEC 62056-6-1:—, 6.2.

Alarm register and alarm filter objects	IC	OBIS code					
		A	B	C	D	E	F
Alarm register objects 1…10	1, Data [a]	0	b	97	98	0…9	255
Alarm register profile object	7, Profile generic	0	b	97	98	255	255
Alarm register table object	61, Register table	0	b	97	98	255	255
Alarm filter objects 1…10	1, Data [a]	0	b	97	98	10…19	255

[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.2.47 General list objects (class_id: 7)

Instances of the IC “Profile generic” are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by “Profile generic” objects. One standard object per billing period scheme is defined.

List objects may be also related to an energy type and to a channel.

General list objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, 6.3 and 7.5.3	7, Profile generic	0	*b*	98	*d*	*e*	255[a]
[a] F = 255 means a wildcard here. See IEC 62056-6-1:—, A.3.							

6.2.48 Event log objects

Instances of the IC “Profile generic” are used to store Event logs. Event logs may be also media related. In this case, the value of value group A shall be the relevant media identifier. See also IEC 62056-6-1:—, 6.5 and 7.5.4.

Event log objects	IC	OBIS code					
		A	B	C	D	E	F
Event log	7, Profile generic	*a*	*b*	99	98	*e*	255[a]
[a] F = 255 means a wildcard here. See IEC 62056-6-1:—, A.3.							
NOTE 1 Event logs can capture for example the time of occurrence of the event, the event code and other relevant data.							
NOTE 2 Companion specifications can specify a more precise meaning of the instances of the different event logs, i.e. the data captured and the number of events captured.							

6.2.49 Inactive objects

Inactive objects are objects, which are present in the meter, but which do not have an assigned functionality. Inactive instances of any IC may be present. See also IEC 62056-6-1:—, 5.3.2.

Inactive objects	IC	OBIS code					
		A	B	C	D	E	F
Inactive objects	Any	0	*b*	127	0	*e*	255

6.3 Electricity related COSEM objects

6.3.1 Value group D definitions

The different ways of processing measurement values as defined by value group D – see IEC 62056-6-1:—, 7.2.1 – are modelled as shown in Table 13.

Table 13 – Representation of various values by appropriate ICs

Type of value	Represented by
cumulative values	Instances of IC "Register" or “Extended register”.
maximum and minimum values	Instances of IC "Profile generic" with sorting method *maximum* or *minimum*, depth according to implementation and captured objects according to implementation. A single maximum value or minimum value can alternatively be represented by an instance of the IC “Register” or "Extended register”.
current and last average values	Respective attributes of instances of the "Demand register", using the OBIS code of the current average value as logical name.
instantaneous values	Instances of the IC "Register”.
time integral values	Instances of the IC "Register” or “Extended register”.
occurrence counters	Instances of the IC “Data” or "Register".
contracted values	Instances of the IC "Register" or “Extended register”.

6.3.2 Electricity ID numbers

The different electricity ID numbers are held by instances of the IC "Data", with data type *octet-string*. If more than one of those is used, it is allowed to combine them into a "Profile generic" object. In this case, the captured objects are *value* attributes of electricity ID "Data" objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, a "Register table" object can be used. See also IEC 62056-6-1:—, Table 19.

Electricity ID objects	IC	OBIS code					
		A	B	C	D	E	F
Electricity ID 1…10 object	1, Data [a]	1	*b*	0	0	0…9	255
Electricity ID-s object	7, Profile generic	1	*b*	0	0	255	255
Electricity ID-s object	61, Register table	1	*b*	0	0	255	255

[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.3.3 Billing period values / reset counter entries

These values are represented by instances of the IC "Data" with data type *unsigned* resp. *octet-string*, formatted as *date_time* in 4.6.1. When the values of historical periods are represented by "Profile generic" objects, the time stamp of the billing period objects shall be part of the captured objects. The values can also be related to a channel.

Billing period values / reset counter entries	IC	OBIS code					
		A	B	C	D	E	F
For item names and OBIS codes see IEC 62056-6-1:—, Table 19.	1, Data [a]	1	*b*	0	1	*e*	255

[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.3.4 Other electricity related general purpose objects

Program entries shall be represented by instances of the IC "Data" with data type *unsigned, long-unsigned* or *octet-string*. For "Meter connection diagram ID" objects data type *enumerated* can be used as well. Program entries can also be related to a channel.

Output pulse constant, reading factor, CT/VT ratio, nominal value, input pulse constant, transformer and line loss coefficient values shall be represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed.

Measurement period, recording interval and billing period duration values shall be represented by instances of IC "Data", "Register" or "Extended register" with the data type of the *value* attribute *unsigned, long-unsigned* or *double-long-unsigned.* The default unit is the second.

Time entry values shall be represented by instances of IC "Data", "Register" or "Extended register" with the data type of the *value* attribute *octet-string*, formatted as *date_time* in 4.6.1. The clock synchronization method shall be represented by an instance of an IC "Data" with data type *enum*.

Synchronization method enum: (0) no synchronization,
(1) adjust to quarter,
(2) adjust to measuring period,
(3) adjust to minute,
(4) reserved,

(5) adjust to preset time,

(6) shift time

For detailed OBIS codes, see IEC 62056-6-1:—, Table 19.

Electricity related general purpose objects	IC	OBIS code					
		A	B	C	D	E	F
Program entries	1, Data [a]	1	*b*	0	2	*e*	255
Output pulse values or constants		1	*b*	0	3	*e*	255
Reading factor and CT/VT ratio		1	*b*	0	4	*e*	255
Nominal values		1	*b*	0	6	*e*	255
Input pulse values or constants		1	*b*	0	7	*e*	255
Measurement period- / recording interval- / billing period duration		1	*b*	0	8	*e*	255
Time entries		1	*b*	0	9	*e*	255
Transformer and line loss coefficients		1	*b*	0	10	*e*	255
[a] If the IC "Data" is not available, the IC "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.							

6.3.5 Measurement algorithm

These values are represented by instances of the IC "Data", with data type *enum*.

Measurement algorithm objects	IC	OBIS code					
		A	B	C	D	E	F
Measurement algorithm for active power	1, Data [a]	1	*b*	0	11	1	255
Measurement algorithm for active energy		1	*b*	0	11	2	255
Measurement algorithm for reactive power		1	*b*	0	11	3	255
Measurement algorithm for reactive energy		1	*b*	0	11	4	255
Measurement algorithm for apparent power		1	*b*	0	11	5	255
Measurement algorithm for apparent energy		1	*b*	0	11	6	255
Measurement algorithm for power factor calculation		1	*b*	0	11	7	255
[a] In cases where the IC "Data" is not available, the IC "Register" (with scaler = 0, unit = 255) may be used.							

The enumerated values are specified in Table 14 below:

Table 14 – Measuring algorithms – enumerated values

Measuring algorithm for active power and energy	
(0)	not specified
(1)	only the fundamentals of voltage and current are used
(2)	all harmonics of voltage and current are used
(3)	only the DC part of voltage and current is used
(4)	all harmonics and the DC part of voltage and current are used
Measuring algorithm for reactive power and energy	
(0)	not specified
(1)	(sum of) reactive power of each phase, calculated from the fundamental of the per phase voltage and the per phase current
(2)	polyphase reactive power calculated from polyphase apparent power and polyphase active power
(3)	(sum of) reactive power calculated from per phase apparent power and per phase active power

Measurement algorithm for apparent power and energy	
(0)	not specified
(1)	$S = U \times I$, with voltage: only fundamental, and current: only fundamental
(2)	$S = U \times I$, with voltage: only fundamental, and current: all harmonics
(3)	$S = U \times I$, with voltage: only fundamental, and current: all harmonics and DC part
(4)	$S = U \times I$, with voltage: all harmonics, and current: only fundamental
(5)	$S = U \times I$, with voltage: all harmonics, and current: all harmonics
(6)	$S = U \times I$, with voltage: all harmonics, and current: all harmonics and DC part
(7)	$S = U \times I$, with voltage: all harmonics and DC part, and current: only fundamental
(8)	$S = U \times I$, with voltage: all harmonics and DC part, and current: all harmonics
(9)	$S = U \times I$, with voltage: all harmonics and DC part, and current: all harmonics and DC part
(10)	$S = \sqrt{P^2 + Q^2}$, with P: only fundamental in U and I, and Q: only fundamental in U and I, where P and Q are polyphase quantities
(11)	$S = \sqrt{P^2 + Q^2}$, with P: all harmonics in U and I, and Q: only fundamental in U and I where P and Q are polyphase quantities
(12)	$S = \sqrt{P^2 + Q^2}$, with P: all harmonics and DC part in U and I, and Q: only fundamental in U and I where P and Q are polyphase quantities
(13)	$S = \Sigma\sqrt{P^2 + Q^2}$, with P: only fundamental in U and I, and Q: only fundamental in U and I where P and Q are single phase quantities
(14)	$S = \Sigma\sqrt{P^2 + Q^2}$, with P: all harmonics in U and I, and Q: only fundamental in U and I where P and Q are single-phase quantities
(15)	$S = \Sigma\sqrt{P^2 + Q^2}$, with P: all harmonics and DC part in U and I, and Q: only fundamental in U and I where P and Q are single-phase quantities
Measurement algorithm for power factor calculation	
(0)	not specified
(1)	displacement power factor: the displacement between fundamental voltage and current vectors, which can be calculated directly from fundamental active power and apparent power, or another appropriate algorithm
(2)	true power factor, the power factor produced by the voltage and current, including their harmonics . It may be calculated from apparent power and active power, including the harmonics.

6.3.6 Metering point ID (electricity related)

A series of objects are available to hold electricity related metering point IDs. They are held by the *value* attribute of “Data” objects, with data type *octet-string*. If more than one of those is used, it is allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are the *value* attributes of the electricity related metering point ID “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC “Register table” can be used. For detailed OBIS codes, see IEC 62056-6-1:—, Table 19.

Metering point ID objects	IC	OBIS code					
		A	B	C	D	E	F
Metering point ID 1…10 (electricity related)	1, Data [a]	1	*b*	96	1	0…9	255
Metering point ID-s object	7, Profile generic	1	*b*	96	1	255	255
Metering point ID-s object	64, Register table	1	*b*	96	1	255	255

[a] If the IC “Data” is not available, the IC “Register” (with scaler = 0, unit = 255) may be used.

6.3.7 Electricity related status objects

A number of electricity related objects are available to hold information about the internal operating status, the starting of the meter and the status of voltage and current circuits.

The status is held by the value attribute of a “Data” object, with data type *bit-string, unsigned, long-unsigned, double-long-unsigned, long64-unsigned or octet-string.*

Alternatively, the status is held by a “Status mapping” object, which holds both the status word and the mapping of its bits to the reference table.

If there are several electricity related internal operating status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global internal operating status. For detailed OBIS codes, see IEC 62056-6-1:—, Table 19.

Electricity related status objects	IC	OBIS code					
		A	B	C	D	E	F
Internal operating status signals, electricity related, contents manufacturer specific	1, Data [a]	1	*b*	96	5	0…5	255
Internal operating status signals, electricity related, contents mapped to reference table	63, Status mapping	1	*b*	96	5	0…5	255
Electricity related status data, contents manufacturer specific	1, Data [a]	1	*b*	96	10	0…3	255
Electricity related status data, contents mapped to reference table	63, Status mapping	1	*b*	96	10	0…3	255
[a] If the IC “Data” is not available, the IC “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

6.3.8 Electricity related list objects (class_id: 7)

These COSEM objects are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by “Profile generic” objects.

One standard object per billing period scheme is defined. See also IEC 62056-6-1:—, 7.5.3.

Electricity related list objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:—, Table 21.	7, Profile generic	1	*b*	98	*d*	*e*	255[a]
[a] F = 255 means a wildcard here. See IEC 62056-6-1:—, A.3.							

6.3.9 Threshold values

A number of objects are available for representing thresholds for instantaneous quantities. The thresholds may be “under limit”, “over limit”, “missing” and “time thresholds”. Time thresholds are used to detect “under limit”, “over limit” and “missing” conditions.

Objects are also available to represent the number of occurrences when these thresholds are exceeded, the duration of such events and the magnitude of the quantity during such events.

These values are represented by instances of IC “Data”, “Register” or “Extended register”.

All these quantities may be related to tariffs.

As defined in IEC 62056-6-1:—, 7.4.2, value group F may be used to identify multiple thresholds.

For OBIS codes, see Table 15 below and IEC 62056-6-1:—, Table 13.

Table 15 – Threshold objects, electricity

Threshold objects	IC	OBIS code					
		A	B	C	D	E	F
Threshold objects for instantaneous values	1, Data, 3, Register, 4, Extended register	1	*b*	1...10, 13, 14, 16...20, 21...30, 33, 34, 36...40, 41...50, 53, 54, 56...60, 61...70, 73, 74, 76...80, 82, 84...89	31...34, 35...38, 39...42, 43...45	0...63	0...99. 255
Threshold objects for harmonics of voltage, current and active power		1	*b*	11, 12, 15, 31, 32, 35, 51, 52, 55, 71, 72, 75, 90...92		0...120, 124... 127	

For monitoring the supply voltage, a more sophisticated functionality is also available, allowing to count the number of occurrences classified by duration of event and depth of voltage dip. For OBIS codes, see IEC 62056-6-1:—, 7.3.6, Table 18.

6.3.10 Register monitor objects (class_id: 21)

Further to 6.2.12, the following applies:

- For monitoring instantaneous values, the logical name of the “Register monitor” object may be the OBIS identifier of the threshold.
- For monitoring current average and last average values, the logical name of the “Register monitor” object may be the OBIS identifier of the demand value monitored.

Table 16 – Register monitor objects, electricity

Register monitor objects	IC	OBIS code					
		A	B	C	D	E	F
Register monitor object, instantaneous values	21, Register monitor	1	*b*	1-80, 82, 84... 92	31, 35, 39	*e*	255
Register monitor object, current average and last average values		1	*b*	1-80, 82, 84... 92	4, 5, 14, 15, 24, 25	*e*	255

The use of value group E shall be to identify the tariff rate. If the total quantity (without tariffication) is monitored, then its value shall be 0.

6.4 Coding of OBIS identifications

To identify different instances of the same IC, their logical_name shall be different. In COSEM, the logical_name is taken from the OBIS definition (see IEC 62056-6-1).

OBIS codes are used within the COSEM environment as an *octet-string [6].* Each octet contains the unsigned value of the corresponding OBIS value group, coded without tags.

If a data item is identified by less than six value groups, all unused value groups shall be filled with 255.

Octet 1 contains the binary coded value of A (A = 0, 1, 2, ...9) in the four rightmost bits. The four leftmost bits contain the information on the identification system. The four leftmost bits set to zero indicate that the OBIS identification system (version 1) is used as logical_name.

Identification system used	Four leftmost bits of octet 1 (MSB left)
OBIS, see IEC 62056-6-1	0 0 0 0
Reserved for future use	0 0 0 1 ... 1 1 1 1

Within all value groups, the usage of a certain selection is fully defined, others are reserved for future use.

If in the value groups B to F a value belonging to the manufacturer specific range (see IEC 62056-6-1:—, 4.8) is used, then the whole OBIS code shall be considered as manufacturer specific, and the value of the other groups does not necessarily carry a meaning defined neither by Clause 5 of this standard nor by IEC 62056-6-1.

7 Previous versions of interface classes

7.1 General

This Clause 7 lists those IC definitions which where included in previous editions of this standard. The previous IC versions differ from the current versions by at least one attribute and/or method and by the version number.

For new implementations in metering devices, only the current versions should be used.

Communication drivers at the client side shall also support previous versions.

7.2 Profile generic (class_id: 7, version: 0)

Profile generic		0...n	class_id = 7, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. buffer	(dyn.)	array			X	x + 0x08
3. capture_objects	(static)	array				x + 0x10
4. capture_period	(static)	double-long-unsigned			X	x + 0x18
5. sort_method	(static)	enum			X	x + 0x20
6. sort_object	(static)	ObjectDefinition			X	x + 0x28
7. entries_in_use	(dyn.)	double-long-unsigned	0		0	x + 0x30
8. profile_entries	(static)	double-long-unsigned	1		1	x + 0x38
Specific methods		***m/o***				
1. reset ()						x + 0x58
2. capture ()						x + 0x60
3. get_buffer_by_range ()						x + 0x68
4. get_buffer_by_index ()						x + 0x70

Attribute description

buffer	The buffer attribute contains a sequence of entries. Each entry contains values of the captured objects (as returned by calling *read(current_value)*). The sequence is ordered according to the specified sort method. The buffer gets filled by subsequent calls to *capture ()*. array Entry Entry::= structure *Instance Specific* Def The buffer is empty at installation. Remark: Reading the entire buffer delivers only those entries which are “in use”.

capture_objects	Specifies the list of capture objects (registers, clocks and profiles) that are assigned to this profile object. Upon a call of the *capture ()* service, the specified attributes of these objects are copied into the buffer of the profile.
	
	array ObjectDefinition
	
	ObjectDefinition::= structure
	{
	logical_name: octet-string;
	class_id: long-unsigned;
	attribute_index: unsigned
	}
	where attribute_index is a pointer to the attribute within the object. attribute_index = 1 refers to the first attribute (i.e. logical_name), attribute_index = 2 to the 2nd, etc.

capture_period	>= 1: Automatic capturing assumed. Specifies the capturing period in seconds 0: no automatic capturing, capturing is 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, 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.

	ObjectDefinition	see above
	Def	no object to sort by (only possible with sort_method= fifo or lifo)
entries_in_use	Counts the number of entries stored in the buffer. After a call of *reset ()* the buffer does not contain any entries, and this value is zero. Upon each subsequent call of *capture ()*, this value will be incremented up to the maximum number of entries that will get stored (see *profile_entries*).	
	double-long-unsigned	0…profile_entries
	Def	0
profile_entries	Specifies how many entries should be retained in the buffer.	
	double-long-unsigned	1… (limited by physical size)
	Def	1

Method description

reset (data)	Clears the buffer. The buffer has no valid entries afterwards, *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 calling *read(<object_attribute>)* of each capture object. Depending on the sort_method and the actual state of the buffer this produces a new entry or a replacement for the less significant entry. As long as all entries have not already been 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)
write (...)	Any write access to one of the attributes describing the static structure of the buffer will automatically call a *reset ()* and this call will propagate to all other profiles capturing this profile. If writing to *profile_entries* is attempted with a value too large for the buffer, it will be set to the maximum possible value (restricted by physical size, typically laid down in the firmware).
get_buffer_by_range (data)	Reads all entries between the given limits. restricting_object in ObjectDefinition Defines the register or clock restricting the range of entries to be retrieved.

	from_value	in	instance_specific Oldest or smallest entry to retrieve.
	to_value	in	instance_specific Newest or largest entry to retrieve.
	selected_values	in	List of columns to retrieve: array ObjectDefinition If the array is empty (has no entries), all captured data is returned. Otherwise, only the columns specified in the array are returned. The type *ObjectDefinition* is specified above (*Capture_Objects*)
	entries	out	See *entries_in_use* above.
	array (of *instance_specific_value)*	out	Sorted sequence of entries containing the requested values of the capture objects.
	data = structure {restricting_object; from_value; to_value; selected_values}. In the response, “data” is an array of instance_specific_value.		
get_buffer_by_index (data)	Read all entries between the given limits.		
	from_index	in	double-long-unsigned First entry to retrieve.
	to_index	in	double-long-unsigned Last entry to retrieve.
	from_selected_value	in	long-unsigned index of the first value to retrieve
	to_selected_value	in	long-unsigned Index of the last value to retrieve.
	entries	out	See *entries_in_use* above.
	array of *instance_specific_value*	out	Sorted sequence of entries containing the requested values of the capture objects.
	data = structure {from_index; to_index; selected_values}. In the response, “data” is an array of instance_specific_value.		

7.3 Association SN (class_id: 12, version: 0)

Device Association View		0..1[10]	class_id = 12, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. object_list	(static)	Object_list_type				x + 0x08
Specific methods		***m/o***				
1. getlist_by_classid		O				x + 0x20
2. getobj_by_logicalname		O				x + 0x28
3. read_by_logicalname()		O				x + 0x30
4. get_attributes&services()		O				x + 0x38
5. change_LLS_secret()		O				x + 0x40
6. change_HLS_secret()		O				x + 0x48
7. get_HLS_challenge()		O				x + 0x50
8. reply_to_HLS_challenge()		O				x + 0x58

Attribute description

logical_name	Identifies the type of the client-server association.

object_list	Contains the list of all objects with their short_name (DLMS ObjectName of the first attribute), class_id, version[11] and logical_name.
	Object_list_type::= array object_list_element
	Object_list_element::= structure
	{
	short_name: Integer16,
	class_id: long-unsigned,
	version: unsigned
	logical_name: octet-string
	}

Method description

getlist_by_classid(data)	Delivers the subset of the object_list for a specific class_id. data::= class_id: long-unsigned For the response: data::= object_list_type

[10] Per client server association

[11] Within n client-server association, there are never two objects with the same class_id and logical_name differing only in the versions.

getobj_by_logicalname (data)	Delivers the entry of the object_list for a specific logical_name and class_id.
	
	data::= structure
	{
	class_id: long-unsigned,
	logical_name: octet-string
	}
	For the response: data::= object_list_element

read_by_logicalname (data)	Reads attributes for specific objects. The objects are specified by their class_id and their logical_name.
	
	data::= array AttributeIdentification
	
	AttributeIdentification::= structure
	
	{
	class_id: long-unsigned,
	logical_name: octet-string,
	attribute_index: unsigned
	}
	where attribute_index is a pointer (i. e. offset) to the attribute within the object.
	
	attribute_index = 0 delivers all attributes[12], attribute_index = 1 delivers the first attribute (i. e. logical_name), etc.
	
	For the response: data is according to the type of the attribute.

[12] If at least one attribute has no read access right under the current association, then a read_by_logicalname() to attribute index = 0 reveals the error message “scope-of-access-violated” (see IEC 61334-4-41:1996, p. 221).

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

change_LLS_secret (data)	Changes the LLS secret (for example password). data::= octet-string new LLS secret
change_HLS_secret (data)	Changes the HLS secret (for example encryption key). data::= octet-string[13] new HLS secret
get_HLS_challenge (data)	Asks the server for the client “challenge” (for example random number). data::= octet-string client challenge
reply_to_HLS_challenge (data)	Delivers the “secretly” processed “challenge” back to the server. data::= octet-string client’s response to the challenge If the authentication is accepted, then the response is successful [0]. If the authentication is not accepted, then the response is set to data-access-error [1].

7.4 Association SN (class_id: 12, version: 1)

This IC allows modelling of AAs between a server and a client, using the application context *short name referencing*. A COSEM logical device may have one instance of this IC for each association the device is able to support.

---

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

The **short_name** of the current "Association SN" object itself is fixed within the COSEM context. It is given in 4.3 as 0x**FA00**.

Association SN	**0...n**	**class_id = 12, version = 1**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. object_list (static)	Object_list_type				x + 0x08
Specific methods	***m/o***				
1. reserved from previous versions	o				
2. reserved from previous versions	o				
3. read_by_logicalname (data)	o				x + 0x30
4. get_attributes&methods (data)	o				x + 0x38
5. change_LLS_secret (data)	o				x + 0x40
6. change_HLS_secret (data)	o				x + 0x48
7. reserved from previous versions					
8. reply_to_HLS_authentication (data)	o				x + 0x58

Attribute description

logical_name Identifies the "Association SN" object instance. See 6.2.22.

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).

```
Object_list_type::=     array object_list_element

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

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

Parameters for selective access to the object_list attribute

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

Method description	

read_by_ logicalname (data)	Reads attributes for selected objects. The objects are specified by their class_id and their logical_name. With this method, the parameterized access feature can also be used.
	
	data::= arrayattribute_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. With this method, the parameterized access feature can also be used.
	
	data::= array object_identification
	
	object_identification::= structure
	{
	class_id: long-unsigned,
	logical_name: octet-string
	}
	For the response
	
	data::= array access_description
	
	access_description::= structure
	{
	read_attributes: bit-string,
	write_attributes: bit-string,
	methods: bit-string
	}
	The position in the bit-string identifies the attribute/method (first position ↔ first attribute, first position ↔ first method) and the value of the bit specifies whether the attribute/method is available (bit set) or not available (bit clear).
	
	Depending on the implementation, some attributes or methods of some objects may not be needed. In this case, such attributes or methods may not be accessible (neither read access, nor write access to attributes, no access to methods).

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

[a] If at least one attribute has no read access right under the current association, then a read_by_logicalname() to attribute index 0 reveals the error message "scope of access violation" (see IEC 61334-4-41:1996, p. 221).

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

7.5 Association LN (class_id: 15, version: 0)

This IC allows modelling of AAs between a server and a client, using the application context *logical name referencing*. Each AA is modelled by one Association LN object.

Association LN	**0...MaxNbofAss.**	**class_id = 15, version = 0**			
Attributes	***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name (static)	octet-string				x
2. object_list (static)	object_list_type				x + 0x08
3. associated_partners_id	associated_partners_type				x + 0x10
4. application_context_name	application-context-name				x + 0x18
5. xDLMS_context_info	xDLMS-context-type				x + 0x20
6. authentication_mechanism_name	mechanism-name				x + 0x28
7. LLS_secret	octet-string				x + 0x30
8. association_status	enum				x + 0x38
Specific methods	***m/o***				
1. reply_to_HLS_authentication (data)	O				x + 0x60
2. change_HLS_secret (data)	O				x + 0x68
3. add_object (data)	O				x + 0x70
4. remove_object (data)	O				x + 0x78

Attribute description

logical_name	Identifies the "Association LN" object instance. See 6.2.22.

object_list

Contains the list of visible COSEM objects with their class_id, version, logical name and the access rights to their attributes and methods within the given application association.

```
object_list_type::= array            object_list_element

object_list_element::= structure

{
      class_id:              long-unsigned,
      version:               unsigned,
      logical_name:          octet-string,
      access_rights:         access_right
}
access_right::= structure
{
      attribute_access:      attribute_access_descriptor,
      method_access:         method_access_descriptor
}
attribute_access_descriptor::= array     attribute_access_item
attribute_access_item::= structure
{
      attribute_id:          integer,
      access_mode:           enum
            {
                  no_access      (0),
                  read_only      (1),
                  write_only     (2),
                  read_and_write (3)
            }
      access_selectors:      CHOICE
            {
                  null-data,
                  array of integer8
            }
}
method_access_descriptor::= arraymethod_access_item
method_access_item::=        structure
{
      method_id:       integer,
      access_mode:           boolean
}
```

where:

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

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

associated_partners_id

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

```
associated_partners_type::= structure
{
      client_SAP:      integer,
      server_SAP:      long-unsigned
}
```

The range for the client_SAP is 0…0x7F.

The range for the server_SAP is 0x000…0x3FFF.

application_context_name

In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that association.

```
CHOICE
{
        structure    [2],
        octet-string [9]
}
```

The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:—, 7.2.2.2.

The application_context_name attribute includes the arc labels of the OBJECT IDENTIFIER. In this case:

```
application_context_name::= structure
        {
        joint-iso-ctt-element:              unsigned,
        country-element:                    unsigned,
        country-name-element:               long-unsigned,
        identified-organization-element:    unsigned,
        DLMS-UA-element:                    unsigned,
        application-context-element:        unsigned,
        context-id-element:                 unsigned
        }
```

For existing implementations, the attribute may hold the value of the OBJECT IDENTIFIER encoded in BER, as an octet string. See IEC 62056-5-3:—, B.4. In this case:

```
application_context_name::= octet-string

// holds the value of the OBJECT identifier encoded in BER
```

EXAMPLE In the case of context_id(1) the A-XDR encoding as a structure (all values are hexadecimal): 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01

The A-XDR encoding as an octet-string, holding the value of the OBJECT IDENTIFIER encoded in BER (all values are hexadecimal): 09 07 60 85 74 05 08 01 01

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
}
```

where:

- the conformance element contains the xDLMS conformance block

supported by the server;

- the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the DLMS-Initiate.response pdu (see IEC 62056-5-3:—, Clause 8).
- the max_send_pdu_size, in an active association contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the DLMS-Initiate.request pdu (see IEC 62056-5-3:—, Clause 8.
- 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 62056-5-3:—, Clause 8.

authentication_ mechanism_name

Contains the name of the authentication mechanism for the association.

```
CHOICE
{
        structure    [2],
        octet-string [9]
}
```

The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-3:—, 7.2.2.3.

The authentication_mechanism_name attribute includes the arc labels of the OBJECT IDENTIFIER. In this case:

```
authentication_mechanism_name::= structure
        {
            joint-iso-ctt-element:                  unsigned,
            country-element:                        unsigned,
            country-name-element:                   long-unsigned,
            identified-organization-element:        unsigned,
            DLMS-UA-element:                        unsigned,
            authentication-mechanism-name-element:  unsigned,
            mechanism-id-element:                   unsigned
        }
```

For existing implementations, the attribute may hold the value of the OBJECT IDENTIFIER encoded in BER, as an octet string. See IEC 62056-5-3:—, B.4. In this case:

```
authentication_mechanism_name::= octet-string

// holds the value of the OBJECT identifier encoded in BER
```

EXAMPLE In the case of mechanism_id(1) the A-XDR encoding as a structure (all values are hexadecimal): 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01

The A-XDR encoding as an octet-string, holding the value of the OBJECT IDENTIFIER encoded in BER (all values are hexadecimal): 09 07 60 85 74 05 08 02 01

	No mechanism-name is required when no authentication is used.
LLS_secret	Contains the authentication value for the LLS authentication process.
association_status	Indicates the current status of the association, which is modelled by the object. enum: (0) non-associated, (1) association-pending, (2) associated

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

Parameters for selective access to the object_list attribute

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

Access selector	Access parameter	Comment
1	NULL	All information excluding the access_rights shall be included in the response.
2	class_list	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
		object_Id::= structure
		{
		class_id: long-unsigned,
		logical_name: octet-string
		}

4	object_Id	The full information record of the required COSEM object instance shall be returned. object_id: See above.

Method description

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

	data::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.
change_HLS_secret (data)	Changes the HLS secret (for example encryption key). data::= octet-string[a] new HLS secret
add_object (data)	Adds the referenced object to the object_list. data::= object_list_element (see above)
remove_object (data)	Removes the referenced object from the object_list. data::= object_list_element (see above)

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

7.6 IEC local port setup (class_id: 19, version: 0)

Instances of this IC define the operational parameters for communication using IEC 62056-21. Several ports can be configured. The logical_names shall be as defined in 6.2.14.

IEC local port setup		**0...n**	**class_id = 19, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. default_mode	(static)	enum				x + 0x08
3. default_baud	(static)	enum				x + 0x10
4. prop_baud	(static)	enum				x + 0x18
5. response_time	(static)	enum				x + 0x20
6. device_addr	(static)	octet-string				x + 0x28
7. pass_p1	(static)	octet-string				x + 0x30
8. pass_p2	(static)	octet-string				x + 0x38
9. pass_w5	(static)	octet-string				x + 0x40
Specific methods (if required)		***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 IC are not applicable.
default_baud	Defines the baud rate for the opening sequence.

	enum	(0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
prop_baud	Defines the baud rate to be proposed by the meter enum	(0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
response_time	Defines the minimum time between the reception of a request (end of request telegram) and the transmission of the response (begin of response telegram). enum	(0) 20 ms, (1) 200 ms
device_addr	Device address according to IEC 62056-21.	
pass_p1	Password 1 according to IEC 62056-21.	
pass_p2	Password 2 according to IEC 62056-21.	
pass_w5	Password W5 reserved for national applications.	

7.7 IEC HDLC setup, (class_id: 23, version: 0)

An instance of the "IEC HDLC setup" contains all data necessary to set up a communication channel according to IEC 62056-46. Several communication channels can be configured. The logical_names shall be as defined in 6.2.16.

IEC HDLC setup		**0...n**	**class_id = 23, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. window_size_transmit	(static)	unsigned	1	7	1	x + 0x10
4. window_size_receive	(static)	unsigned	1	7	1	x + 0x18
5. max_info_field_length_ transmit	(static)	unsigned	32	128	128	x + 0x20
6. max_info_field_length_ receive	(static)	unsigned	32	128	128	x + 0x28
7. inter_octet_time_out	(static)	long-unsigned	20	1 000	25	x + 0x30
8. inactivity_time_out	(static)	long-unsigned	0		120	x + 0x38

IEC HDLC setup		0...n	class_id = 23, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
9. device_address	(static)	long-unsigned	0x0010	0x3FFD		x + 0x40
Specific methods (if required)		***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 the corresponding station. During logon, other values can be negotiated.

window_size_receive

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

max_info_length_transmit

The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.

max_info_length_receive

The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.

inter_octet_time_out

Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame.

inactivity_time_out

Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection.

When this value is set to 0, this means that the inactivity_time_out is not operational.

device_address Contains the physical device address of a device:

In the case of single byte addressing:

0x00 NO_STATION Address,
0x01…0x0F Reserved for future use,
0x10…0x7D Usable address space,
0x7E 'CALLING' device address,
0x7F Broadcast address

In the case of double byte addressing:

0x0000 NO_STATION address,
0x0001…0x000F Reserved for future use,
0x0010…0x3FFD Usable address space,
0x3FFE 'CALLING' physical device address,
0x3FFF Broadcast address

7.8 PSTN modem configuration (class_id: 27, version: 0)

NOTE The name of this IC was changed to "Modem configuration" in IEC 62056-62:2006.

An instance of the "PSTN modem configuration" IC stores data related to the initialization of modems, which are used for data transfer from/to a device. Several modems can be configured. The logical_names shall be as defined in 6.2.5.

PSTN modem configuration		0…n	class_id = 27, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. initialization_string	(static)	array				x + 0x10
4. modem_profile	(static)	array				x + 0x18
Specific methods		***m/o***				

Attribute description

comm_speed The communication speed between the device and the modem, not necessarily the communication speed on the WAN.

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

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

modem_profile	This data defines the mapping from Hayes standard commands/responses to modem specific strings. array modem_profile_element modem_profile_element::= octet-string The modem_profile array must contain the corresponding strings for the modem used in the 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

7.9 PSTN auto dial (class_id: 29, version: 0)

NOTE The name of this IC was changed to “Auto conect” in IEC 62056-62:2006.

An instance of the “PSTN auto dial” IC stores data related to the management data transfer between the device and the modem to perform auto dialling. Several modems can be configured. The logical_names shall be as defined in 6.2.5.

PSTN auto dial		**0...n**	**class_id = 29, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08

PSTN auto dial		0...n	class_id = 29, version = 0			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
3. repetitions	(static)	unsigned				x + 0x10
4. repetition_delay	(static)	long-unsigned				x + 0x18
5. calling_window	(static)	array				x + 0x20
6. phone_list	(static)	array				x + 0x28
Specific methods		***m/o***				

Attribute description

logical_name

Identifies the “PSTN auto dial” object instance. See 6.2.5.

mode

Defines if the device can perform auto-dialling.0

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 the case of unsuccessful dialling attempts.

repetition_delay

The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated.

repetition_delay 0 means delay is not specified

calling_window

Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date implicitly defines 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.

array window_element

window_element::= structure
{
start_time: octet-string,
end_time: octet-string
}

start_time and end_time are formatted as set in 4.6.1 for *date_time*

phone_list	Contains phone numbers, the device modem has to call under certain conditions. The link between entries in the array and the conditions are not contained in here. array phone_number phone_number::= octet-string

7.10 S-FSK Phy&MAC setup (class_id: 50, version: 0)

NOTE 1 The use of this version is deprecated.

An instance of the "S-FSK Phy&MAC setup" IC stores the data necessary to set up and manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

S-FSK Phy&MAC setup		**0...n**	**class_id = 50, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. initiator_electrical_phase	(static)	enum	0	3		x + 0x08
3. delta_electrical_phase	(dyn.)	enum	0	6		x + 0x10
4. max_receiving_gain	(static)	unsigned				x + 0x18
5. max_transmitting_gain	(static)	unsigned				x + 0x20
6. search_initiator_gain	(static)	unsigned				x + 0x28
7. frequencies	(static)	frequencies_type				x + 0x30
8. mac_address	(dyn.)	long-unsigned			FFE	x + 0x38
9. mac_group_addresses	(static)	array				x + 0x40
10. repeater	(static)	enum			1	x + 0x48
11. repeater_status	(dyn.)	boolean				x + 0x50
12. min_delta_credit	(dyn.)	unsigned				x + 0x58
13. initiator_mac_address	(dyn.)	long-unsigned				x + 0x60
14. synchronization_locked	(dyn.)	boolean				x + 0x68
Specific methods		***m/o***				

Attribute description

logical_name	Identifies the "S-FSK Phy&MAC setup" object instance. See 6.2.20.
initiator_electrical_phase	Holds the MIB variable initiator-electrical-phase (variable 18) specified in IEC 61334-4-512:2001, 5.8. It is written by the client system to indicate the phase to which it is connected. enum: (0) Not defined (default), (1) Phase 1, (2) Phase 2, (3) Phase 3.

NOTE 2 This enumeration is different from that of IEC 61334-4-512.

delta_electrical_phase	Holds the MIB variable delta-electrical-phase (variable 1) specified in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1:2001, 3.5.5.3. It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined: enum: (0) Not defined: the server is temporarily not able to determine the phase difference, (1) The server system is connected to the same phase as the client system. The phase difference between the server's connecting phase and the client's connecting phase is equal to: (2) 60 degrees, (3) 120 degrees, (4) 180 degrees, (5) -120 degrees, (6) -60 degrees.
max_receiving_gain	Holds the MIB variable max-receiving-gain (variable 2) specified in IEC 61334-4-512:2001, 5.2 and in IEC 61334-5-1:2001, 3.5.5.3. Corresponds to the maximum allowed gain bound to be used by the server system in the receiving mode. The default unit is dB. NOTE 3 In IEC 61334-4-512, no units are specified. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
max_transmitting_gain	Holds the value of the max-transmitting-gain. Corresponds to the maximum attenuation bound to be used by the server system in the transmitting mode. The default unit is dB. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
search_initiator_gain	This attribute is used in the intelligent search initiator process. If the value of the max_receiving_gain is below the value of this attribute, a fast synchronization process is possible.

frequencies	Contains frequencies required for S-FSK modulation.
	
	frequencies_type::= structure
	{
	mark_frequency: double-long-unsigned,
	space_frequency: double-long-unsigned
	}
	The default unit is Hz.

mac_address	Holds the MIB variable mac-address (variable 3) specified in

	IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 MAC addresses are expressed on 12 bits. Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is “NEW-address”. This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address". This attribute is set to NEW: • by the MAC sub-layer, once the time-out-not-addressed delay is exceeded; • when a client system “resets” the server system. See the “S-FSK Active initiator” IC in 5.8.5. When this attribute is set to NEW: • the system loses its synchronization (function of the MAC-sub-layer); • the mac_group_address attribute is reset (array of 0 elements); • the system automatically releases all AAs which can be released. NOTE 5 The second item is not present in IEC 61334-4-512. The predefined MAC addresses are shown in Table 11.
mac_group_addresses	Holds the MIB variable mac-group-address (variable 4) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Contains a set of MAC group addresses used for broadcast purposes. array mac-address mac-address::= long-unsigned The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included in this list. These ones are internal predefined values. This attribute shall be written by the initiator using DLMS services to declare specific MAC group addresses on a server system. This attribute is locally read by the MAC sub-layer when checking the destination address field of a MAC frame not recognized as an individual address or as one of the three predefined values (ALL-configured-address, ALL-physical-address and NO-BODY).
repeater	Holds the MIB variable repeater (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. It specifies whether the server system effectively repeats all frames or not.

	enum: (0) never repeater, (1) always repeater, (2) dynamic repeater If the repeater variable is equal to 0, the server system should never repeat the frames. If it is set to 1, the server system is a repeater: it has to repeat all frames received without error and with a current credit greater than zero. If it is set to 2, then the repeater status can be dynamically changed by the server itself. NOTE 6 The value 2 value is not specified in IEC 61334-4-512. This attribute is internally read by the MAC sub-layer each time a frame is received. The default value is 2, and the server system is a repeater (repeater_status = TRUE).
repeater_status	Holds the current repeater status of the device. boolean: FALSE = no repeater, TRUE = repeater
min_delta_credit	Holds the MIB variable min-delta-credit (variable 9) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 7 Only the three least significant bits are used. The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields of a correctly received MAC frame. The delta-credit minimum value of a correctly received MAC frame, directed to a server system, is held by this variable. The default value is set to the maximal initial credit (see IEC 61334-5-1:2001, 4.2.3.1 for further explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can reinitialise this variable by setting its value to the maximal initial credit.
initiator_mac_address	Holds the MIB variable initiator-mac-address specified in IEC 61334-5-1:2001, 4.3.7.6. Its value is either the MAC address of the active-initiator or the NO-BODY address, depending on the value of the synchronization_locked attribute (see below). See also IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2. If the value NO-BODY is written then the server mac_address (see the mac_address attribute) has to be set to NEW.
synchronization_locked	Holds the MIB variable synchronization-locked (variable 10) specified in IEC 61334-4-512:2001, 5.3. Controls the synchronization locked / unlocked state. See IEC 61334-5-1 for more details. If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state. In this state, the initiator-mac-address

is always equal to the MAC address field of the active-initiator MIB object. See attribute 2 of the S-FSK Active initiator IC, 5.8.5.

If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the initiator_mac_address attribute is always set to the NO-BODY value: a value change in the MAC address field of the active-initiator MIB object does not affect the content of the initiator_mac_address attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications.

NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of which is equal to the value of the initiator_mac_address attribute.

7.11 S-FSK IEC 61334-4-32 LLC setup (class_id: 55, version: 0)

An instance of the "S-FSK IEC 61334-4-32 LLC setup" IC holds parameters necessary to set up and manage the LLC layer as specified in IEC 61334-4-32.

S-FSK IEC 61334-4-32 LLC setup		**0...n**	**class_id = 55, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. max_frame_length	(static)	unsigned	26	242	134	x + 0x08
3. reply_status_list	(dyn.)	array				x + 0x10
Specific methods		***m/o***				

Attribute description

logical_name

Identifies the "S-FSK IEC 61334-4-32 LLC setup" object instance. For logical names, see 6.2.20.

max_frame_length

Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4.

In the case of the S-FSK profile, as specified in IEC 61334-5-1:2001, 4.2.2, the maximum value is 242, but lower values may be chosen due to performance considerations.

reply_status_list	Holds the MIB variable *reply-status-list* (variable 11) specified in IEC 61334-4-512:2001, 5.4.
	
	Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not already been read. The length of a waiting L-SDU is specified in number of sub-frames (different from zero). The variable is locally generated by the LLC sub-layer.
	
	array reply_status
	
	reply_status::= structure
	{
	L-SAP-selector: unsigned,
	length-of-waiting-L-SDU: unsigned
	}
	length-of-waiting-LSDU in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.

Annex A
(informative)

Significant technical changes with respect to IEC 62056-62

This part of IEC 62056 includes the following significant technical changes with respect to IEC 62056-62:

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

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

- in 5.2.2 Table 3 new quantities have been added: 65 *Specific energy* and 70 *Signal strength*;
- in 5.2.2 Table 4 new examples have been added;
- 5.3.1 defines a new version of the Association SN IC: class_id: 12, version: 2;
- 5.3.2 defines a new version of the Association LN: class_id: 15, version: 1;
- 18 new interface classes have been added (IEC 62056-62 contained only 31 (19 in Clause 5 and 12 in Annex A), now there are 49), see the following details:
  - in 5.3.3 *SAP assignment* class for the logical_device_name element, a CHOICE of data types is allowed now;
  - 5.3.4: new Image transfer IC (class_id: 18, version: 0) has been added to support firmware upgrade;

- 5.3.5: new Security setup IC (class_id: 64, version: 0) has been added to support a new HLS authentication mechanism and to support data transport security;
- 5.4.8: new Disconnect control IC (class_id: 70, version: 0) has been added to support the connection and disconnection of the supply;
- 5.4.9: new Limiter IC (class_id: 71, version: 0) has been added to support the limitation of the supply under certain conditions;
- 5.4.10: new Sensor manager interface class (class_id: 67, version: 0) has been added to manage sensors and monitoring their operation. This IC is mainly used with non-electricity meters;
- 5.6.1: new M-bus slave port setup (class_id: 25, version: 0) has been added to allow the meter to be configured as an M-Bus slave device;
- 5.6.2: new M-bus client (class_id: 72, version: 0) has been added to allow the meter to be configured as an M-Bus client device;
- 5.6.3: new Wireless Mode Q channel (class_id: 73, version: 1) has been added;
- 5.6.4: new M-bus master port setup (class_id: 74, version: 0) has been added to configure the M-Bus port;
- in 5.7.1 *TCP-UDP setup* (class_id: 41, version: 0), the assigned port numbers have been added;
- in 5.7.3 the name and the use of “Ethernet setup” interface class has been changed to “MAC address setup” without changing the version;
- in 5.7.6 SMTP setup (class_id: 46, version: 0): port numbers have been added;
- new 5.8 *Interface classes for setting up data exchange using S-FSK PLC* has been added to manage the DLMS/COSEM S-FSK PLC profile:
  - 5.8.4 *S-FSK Phy&MAC set-up* (class_id: 50, version: 1);
  - 5.8.5 *S-FSK Active initiator* (class_id: 51, version: 0);
  - 5.8.6 *S-FSK MAC synchronization timeouts* (class_id: 52, version: 0);
  - 5.8.7 *S-FSK MAC counters* (class_id: 53, version: 0);
  - 5.8.8 *IEC 61334-4-32 LLC setup* (class_id: 55, version: 1);
  - 5.8.9 *S-FSK Reporting system list* (class_id: 56, version: 0);
  - 5.9.3 *ISO/IEC 8802-2 LLC Type 1 setup* (class_id: 57, version: 0);
  - 5.9.4 *ISO/IEC 8802-2 LLC Type 2 setup* (class_id: 58, version: 0);
  - 5.9.5 *ISO/IEC 8802-2 LLC Type 3 setup* (class_id: 59, version: 0).

- Clause 6 *Relation to OBIS* includes now the contents of the former Annex D, completed with the new interface class definitions, OBIS code allocations and data types;
- 6.2.2 *Data of historical billing periods* now explains the rules related to the use of various interface classes and the use of value group F for identifying historical data;
- 6.2.3 *Billing period values / reset counter entries*: abstract entries with A = 0 are also allowed now;
- in 6.2.4, value group E has been changed from “0” to “e” to allow several instances of the Clock object,
- in 6.2.6 *Script table objects* (class_id: 9) new instances have been specified;
- in 6.2.11 Newinstances have been specified;
- 6.2.25 specifies Security setup and frame counter objects;
- 6.2.40 specifies Event code objects;
- 6.2.42 specifies Consumer message objects;
- 6.2.43 specifies Currently active tariff objects;
- 6.2.44 specifies Event counter objects;

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

Bibliography

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

EN 13757-1:2002, *Communication system for meters and remote reading of meters – Part 1: Data exchange*

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

IEC 61334-6:2000, *Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule*

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

IEC 62056-8-3:—, *Electricity metering data exchange – The DLMS/COSEM suite – Part 8-3: Communication profile for PLC S-FSK neighbourhood networks*[14]

DLMS UA 1000-1:2010, *COSEM Identification System and Interface Classes, the "Blue Book"*

DLMS UA 1000-2:2009, *DLMS/COSEM Architecture and Protocols, the "Green Book"*

DLMS UA 1001-1:2010, *DLMS/COSEM Conformance Test and certification process, the "Yellow Book"*

ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 15953:1999, *Information technology – Open Systems Interconnection – Service definition for the Application Service Object Association Control Service Element*

ISO/IEC 15954:1999, *Information technology – Open Systems Interconnection – Connection-mode protocol for the Application Service Object Association Control Service Element*

ISO/IEC 8824-1:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8825-1:2008, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 13239:2002, *Information Technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures*

ITU Recommendation X.217:1995, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element*

14 To be published simultaneously with this part of IEC 62056.

ITU Recommendation X.227:1995, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the Association Control Service Element: Protocol specification*

IEEE 754:1985, *IEEE Standard for Binary Floating-Point Arithmetic*

ISO/IEC 10646:2012, *Information technology – Universal Coded Character Set (UCS)*

FIPS PUB 180-1:2002, *Secure hash standard*

FIPS PUB 197:2001, *Advanced Encryption Standard (AES)*

FIPS PUB 198:2002, *The Keyed-Hash Message Authentication Code (HMAC)*

FIPS PUB 199:2002, *Standards for Security Categorization of Federal Information and Information Systems*

NIST SP 800-21:2005, *Guideline for Implementing Cryptography in the Federal Government*

NIST SP 800-38D:2007, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*

NIST SP 800-57:2006, *Recommendation for Key Management – Part 1: General* (Revised)

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

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

Index

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

ÉCHANGE DE DONNÉES DANS LES ÉQUIPEMENTS DE COMPTAGE DE L'ÉNERGIE ÉLECTRIQUE – LA SUITE DLMS/COSEM –

Partie 6-2: Classes d'interfaces COSEM

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Commission Électrotechnique Internationale (CEI) attire l'attention sur le fait qu'il est déclaré que la conformité à la présente Norme internationale peut nécessiter l'utilisation d'un service de maintenance concernant la pile de protocoles sur laquelle est basée la présente Norme CEI 62056-6-2.

La CEI ne prend pas position concernant la preuve, la validité et le domaine d'application de ce service de maintenance.

Le fournisseur du service de maintenance a assuré à la CEI qu'il souhaite fournir des services aux demandeurs dans le monde entier, selon des termes et les conditions raisonnables et non discriminatoires. À cet égard, la déclaration du fournisseur du service de maintenance est enregistrée avec la CEI. Des informations peuvent être obtenues auprès de:

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

La Norme internationale CEI 62056-6-2 a été établie par le comité d'études 13 de la CEI: Mesure de l'énergie électrique, contrôle des tarifs et de la charge.

Cette édition annule et remplace la CEI 62056-62, parue en 2006. Cette édition constitue une révision technique.

Les modifications techniques majeures par rapport à la CEI 62056-62 sont énumérées dans l'Annexe A.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
13/1525/FDIS	13/1543/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 62056, publiées sous le titre général *Échange de données dans les équipements de comptage de l'énergie électrique – La suite DLMS/COSEM*, peut être consultée sur le site web de la CEI.

Les futures normes de cette série porteront dorénavant le nouveau titre général cité ci-dessus. Le titre des normes existant déjà dans cette série sera mis à jour lors de la prochaine édition.

La numérotation est passée de CEI 62056-XY à CEI 62056-X-Y. Par exemple, la CEI 62056-62 devient la CEI 62056-6-2.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

[1] Spécification de message de langage de dispositif.

INTRODUCTION

Dicté non seulement par les besoins des fournisseurs de service d'électricité – mais aussi souvent dans un marché compétitif dérégulé – mais aussi par le souhait de plus en plus affirmé de gérer efficacement les ressources naturelles en termes de production, de distribution et d'utilisation, le compteur d'énergie électrique prend une part croissante dans les systèmes intégrés de comptage, commande et facturation. Non seulement au niveau du réseau électrique mais aussi, avec l'avènement d'initiatives pour impliquer les consommateurs dans la gestion de l'énergie et des ressources, que ce soit au niveau industriel ou au niveau domestique, le compteur d'énergie n'est pas un simple dispositif d'enregistrement de données mais s'appuie de façon critique sur les capacités de communication, l'intégration des systèmes et l'interopérabilité.

La COSEM (Companion Specification for Energy Metering), à savoir la spécification d'accompagnement pour le comptage de l'énergie, traite ces défis en considérant le compteur comme une partie intégrante d'un système de communication qui exige par-dessus tout la capacité d'acheminer sur une diversité de supports de connexion les mesures de la livraison (de l'énergie électrique) depuis les divers points où ces mesures sont effectuées jusqu'aux processus commerciaux qui les utilisent. De tels systèmes traitent une gamme d'informations complémentaires et prennent en charge les fonctions d'établissement et de commande qui permettent d'exploiter le compteur à distance, pratiquement à tout instant.

La COSEM accomplit tout cela d'une manière essentiellement non-propriétaire et ne fait pas de suppositions sur les processus techniques en place au sein du compteur. En utilisant des techniques de *modélisation d'objets* établies dans le monde informatique, les données devant être fournies par le compteur sont définies d'une manière normalisée qui est accessible aux processus métier des entreprises de services électriques; les parties pertinentes de leur comportement sont représentées d'une manière semblable, alors que les communications sont définies conformément à l'*Interconnexion des systèmes ouverts (OSI)* qui est fondamentale pour le monde des télécommunications. La spécification formelle des classes d'interfaces et des objets, qui le permet, constitue une partie importante de la COSEM.

Afin de permettre une analyse plus approfondie de l'information, aux fins de la facturation, de la gestion des charges, des clients et des contrats, il est nécessaire d'identifier de façon univoque les éléments de données, qu'ils soient recueillis manuellement ou automatiquement, via l'échange de données local ou distant, d'une manière indépendante du fabricant. La définition des codes d'identification pour ce faire – les codes OBIS – est basée sur la norme DIN 43863-3:1997, *Electricity meters – Part 3: Tariff metering device as additional equipment for electricity meters – EDIS – Energy Data Identification System* (disponible en anglais seulement).

Le modèle COSEM représente le compteur comme un serveur – voir 4.7– utilisé par les applications de client qui récupèrent des données du compteur, fournissent des informations de commande au compteur et déclenchent des actions au sein du compteur via l'accès contrôlé aux attributs et méthodes spécifiques d'objets constituant l'interface au serveur. Ce client peut être engagé dans la prise en charge des processus métier des entreprises d'électricité, des clients, des opérateurs de compteurs ou des fabricants de compteurs.

Le contenu de l'information et les capacités du serveur ne sont pas fixés; par contre, les objets et classes d'interfaces (les IC) normalisés forment une bibliothèque extensible à partir de laquelle le fabricant peut assembler (modéliser) ses produits conformément à des spécifications nationales ou à des exigences contractuelles. En tant qu'élément clé, le serveur offre un moyen de récupérer son modèle particulier de structure (la liste des dispositifs logiques et la liste des objets visibles à travers l'interface). Par sa conception, la bibliothèque permet de couvrir la gamme complète de produits (depuis les applications résidentielles jusqu'aux applications commerciales, industrielles, de transport et de distribution). Le choix du sous-ensemble des IC utilisées pour construire un compteur et aussi l'instanciation et la mise en œuvre de ces IC constituent une partie intégrante de la conception du produit et sont donc laissés au fabricant. Le concept de la bibliothèque normalisée de classes d'interfaces

pour le comptage fournit aux différents utilisateurs et fabricants un maximum de diversité sans sacrifier l'interopérabilité.

ÉCHANGE DE DONNÉES DANS LES ÉQUIPEMENTS DE COMPTAGE DE L'ÉNERGIE ÉLECTRIQUE – LA SUITE DLMS/COSEM –

Partie 6-2: Classes d'interfaces COSEM

1 Domaine d'application

La présente partie de la CEI 62056 spécifie un modèle d'un compteur tel qu'il est vu à travers son/ses interface(s) de communication. Des blocs génériques de base sont définis à l'aide de méthodes orientées objet, sous la forme de classes d'interfaces pour modéliser les compteurs à partir d'une fonctionnalité simple jusqu'à une fonctionnalité très complexe.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 61334-4-32:1996, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4: Protocoles de communication de données – Section 32: Couche liaison de données – Contrôle de liaison logique (LLC)*

CEI 61334-4-41:1996, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4: Protocoles de communication de données – Section 41: Protocoles d'application – Spécification des messages de ligne de distribution*

CEI 61334-4-511:2000, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4-511: Protocoles de communication de données – Administration de systèmes – Protocole CIASE*

CEI 61334-4-512:2001, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4-512: Protocoles de communication de données – Administration de systèmes à l'aide du profil 61334-5-1 – MIB (Base d'Informations d'Administration)*

CEI 61334-5-1:2001, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 5-1: Profils des couches basses – Profil S-FSK (modulation par saut de fréquences étalées)*

CEI/TR 62051:1999, *Lecture des compteurs électriques – Glossaire de termes* (disponible en anglais seulement)

CEI/TR 62051-1:2004, *Electricity metering – Data exchange for meter reading, tariff and load control – Glossary of terms – Part 1: Terms related to data exchange with metering equipment using DLMS/COSEM* (disponible en anglais seulement)

CEI 62056-21:2002, *Équipements de mesure de l'énergie électrique – Échange des données pour la lecture des compteurs, le contrôle des tarifs et de la charge – Partie 21: Échange des données directes en local*

Projet CEI 62056-3-1:—, *Comptage de l'électricité – Echange de données pour la lecture des compteurs, le contrôle des tarifs et de la charge – Partie 3-1: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse*[2]

CEI 62056-46:2002, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol* (disponible en anglais seulement)
Amendement 1:2006

CEI 62056-5-3:—, *Échange de données dans les équipements de comptage de l'énergie électrique – La suite DLMS/COSEM – Partie 5-3: Couche application DLMS/COSEM*[3]

CEI 62056-6-1:—, *Échange de données dans les équipements de comptage de l'énergie électrique – La suite DLMS/COSEM – Partie 6-1: Système d'identification d'objets (OBIS)*[4]

ISO/CEI 8802-2:1998, *Technologies de l'information – Télécommunications et échange d'information entre systèmes – Réseaux locaux et métropolitains – Exigences spécifiques – Partie 2: Contrôle de liaison logique* (disponible en anglais seulement)

ISO/IEC/IEEE 60559:2011, *Information technology – Microprocessor Systems – Floating-Point arithmetic* (disponible en anglais seulement)

EN 13757-2:2004, *Systèmes de communication et de télérelevé de compteurs – Partie 2: Couche physique et couche de liaison*

EN 13757-3:2004, *Systèmes de communication et de télérelevé de compteurs – Partie 3: Couches d'application spéciale*

EN 13757-5:2008, *Lecture sans fil des compteurs – Système de communication et de télérelevé de compteur – Partie 5: Mise en place de relais*

ANSI C12.19:1997, *IEEE 1377.1997, Utility industry end device data tables* (disponible en anglais seulement)

RFC 1332: 1992, Internet Engineering Task Force (IETF). *The PPP Internet Protocol Control Protocol (IPCP)*. Éditée par G. McGregor, mai 1992. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc1332.txt (disponible en anglais seulement)

RFC 1570: 1994, Internet Engineering Task Force (IETF). *PPP LCP Extensions*. Éditée par W. Simpson, janvier 1994. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc1570.txt (disponible en anglais seulement)

RFC 1661: 1994, Internet Engineering Task Force (IETF) *The Point-to-Point Protocol (PPP)* (également: IETF STD 0051), 1994. Éditée par W. Simpson, juillet 1994. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc1661.txt (disponible en anglais seulement)

RFC 1662: 1994, Internet Engineering Task Force (IETF). *PPP in HDLC-like Framing* (également: IETF STD 0051). Éditée par W. Simpson, juillet 1994. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc1662.txt (disponible en anglais seulement)

---

2 À publier.

3 À publier simultanément avec la présente partie de la CEI 62056.

4 À publier simultanément avec la présente partie de la CEI 62056.

RFC 1700:1994, Internet Engineering Task Force (IETF). *Assigned numbers* (également: IETF STD 0002). Éditée par J. Reynolds, J. Postel, octobre 1994. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc1700.txt (disponible en anglais seulement)

RFC 2507:1999, Internet Engineering Task Force (IETF). *IP Header Compression.* Éditée par M. Degermark, B. Nordgren, S. Pink, octobre 1994. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc2507.txt (disponible en anglais seulement)

RFC 3241: 2002, Internet Engineering Task Force (IETF). *Robust Header Compression (ROHC) over PPP, 2002.* Éditée par C. Bormann, avril 2002. Disponible à l'adresse: http://www.rfc-editor.org/rfc/rfc3241.txt (disponible en anglais seulement)

STD 0005:1981, Internet Engineering Task Force (IETF). *Internet Protocol. Également: RFC0791 (RFC0792, RFC0919, RFC0922, RFC0950, RFC1112). Intellectual Property Rights in IETF Technology*. Éditée par Jon Postel. septembre 1981. Disponible à l'adresse: http://www.faqs.org/rfcs/std/std5.html (disponible en anglais seulement)

STD 0051:1994, Internet Engineering Task Force (IETF): *The Point-to-Point Protocol (PPP).* Éditée par W. Simpson juillet 1994. (Également: RFC1661, RFC1662). Disponible à l'adresse: http://www.faqs.org/rfcs/std/std51.html (disponible en anglais seulement)

NOTE Voir aussi la Bibliographie.

3 Abréviations

AA	Application Association (Association d'applications)
AARE	A-Associate Response (Réponse d'association d'applications) – une APDU de l'ACSE
AARQ	A-Associate Request (Demande d'association d'applications) – une APDU de l'ACSE
ACSE	Association Control Service Element (Élément de service de contrôle d'association)
AL	Application Layer (Couche application)
AP	Application process (Processus d'application)
APDU	Application Layer Protocol Data Unit (Unité de données de protocole d'application)
ASE	Application Service Element (Élément de service d'application)
A-XDR	Adapted Extended Data Representation (Représentation de données étendues adaptée)
base_name	Le short_name correspondant au premier attribut (“logical_name”) d'un objet COSEM
CHAP	Challenge Handshake Authentication Protocol (Protocole d'authentification par challenge)
class_id	Code d'identification d'une classe d'interfaces
COSEM	Companion Specification for Energy Metering (Spécification d'accompagnement pour le comptage de l'énergie)
Objet COSEM	Instance d'une classe d'interfaces COSEM
CtoS	Client to Server challenge (Défi du client au serveur)
DHCP	Dynamic Host Configuration Protocol (Protocole de configuration dynamique d'hôte)

DLMS	Device Language Message Specification (Spécification de message de langage de dispositif)
DLMS UA	DLMS User Association
DNS	Domain Name System (Serveur de noms de domaine)
EAP	Extensible Authentication Protocol (Protocole d'authentification extensible)
GCM	Galois/Counter Mode (Mode Galois/Compteur) – un algorithme de chiffrement authentifié avec données associées
GMT	Greenwich Mean Time (Temps moyen de Greenwich) Remplacé par le Temps Universel Coordonné (TUC).
GPS	Global Positioning System (Système de positionnement global)
GPRS	General Packet Radio Service (Service général de radiocommunications en mode paquet)
GSM	Global System for Mobile Communications (Système global de communications mobiles)
HART	Highway Addressable Remote Transducer (Transducteur à distance adressable par bus) voir http://www.hartcomm.org/ (en rapport avec la classe d'interfaces Gestionnaire de capteur)
HDLC	High-level Data Link Control (Commande de liaison de données à haut niveau)
HLS	High Level Security (Sécurité de haut niveau)
IANA	Internet Assigned Numbers Authority (Autorité chargée de l'assignation des numéros Internet)
IC	Interface Class (Classe d'interfaces)
CEI	Commission Électrotechnique Internationale
IEEE	Institute of Electrical and Electronics Engineers (Institut d'ingénieurs en électricité et en électronique)
IETF	Internet Engineering Task Force (Groupe de travail d'ingénierie Internet)
IP	Internet Protocol (Protocole internet)
IPCP	Internet Protocol Control Protocol (Protocole de contrôle du Protocole Internet)
TI	Technologie de l'information
ISO	International Organization for Standardization (Organisation Internationale de Normalisation)
KEK	Key Encryption Key (clé de chiffrement de clé)
LCP	Link Control Protocol (Protocole de commande de liaison)
LLC	Logical Link Control (Commande de liaison logique) – sous-couche
LLS	Low Level Security (Sécurité de bas niveau)
LN	Logical Name (Nom logique) (utilisé en rapport avec les attributs et méthodes de référencement d'objets COSEM)
LSB	Least Significant Bit (Bit de poids faible)
m	mandatory (obligatoire)
MD5	Message Digest Algorithm 5 (Algorithme de condensation de message 5)
MID	Measuring Instruments (Instruments de mesure) Directive 2004/22/CE
MSB	Most Significant Bit (Bit de poids fort)
o	optional (facultatif)

OBIS	OBject Identification System (Système d'identification d'objet)
PAP	Password Authentication Protocol (Protocole d'authentification de mot de passe)
PDU	Protocol Data Unit (Unité de données de protocole)
PLMN	Public Land Mobile Network (Réseau mobile terrestre public)
PIN	Personal Identity Number (Numéro d'identification personnel)
PPP	Point-to-Point Protocol (Protocole point à point)
PSTN	Public Switched Telephone Network (Réseau Téléphonique Public Commuté)
RDR	Reply Data on Request (Données de réponse sur demande) (utilisée dans la CEI 61334-4-32)
REJ PDU	Reject Protocol Data Unit (Unité de données de protocole de rejet)
ROHC	Robust Header Compression (Compression robuste des en-têtes)
SAP	Service Access Point (Point d'accès au service)
SHA-1	Secure Hash Algorithm (Algorithme de hachage sécurisé)
SMS	Short Message Service (Service de messages courts)
SMTP	Simple Mail Transfer Protocol (Protocole simple de transfert de courrier)
SN	Short Name (Nom court) (utilisé en rapport avec les attributs et méthodes de référencement d'objets COSEM)
StoC	Server to Client Challenge (Défi du serveur au client)
TUC	Temps universel coordonné
UI PDU	Unnumbered Information Protocol Data Unit (Unité de données de protocole à information non numérotée)

4 Principes de base

4.1 Généralités

Le présent Article 4 décrit les principes de base sur lesquels sont construites les classes d'interfaces COSEM (les IC). Il donne aussi une courte vue d'ensemble sur la manière d'utiliser les objets d'interfaces – instanciations des IC – à des fins de communication. Les systèmes de collecte de données et les équipements de comptage issus de différents vendeurs, conformes à ces spécifications, peuvent échanger des données d'une manière interopérable.

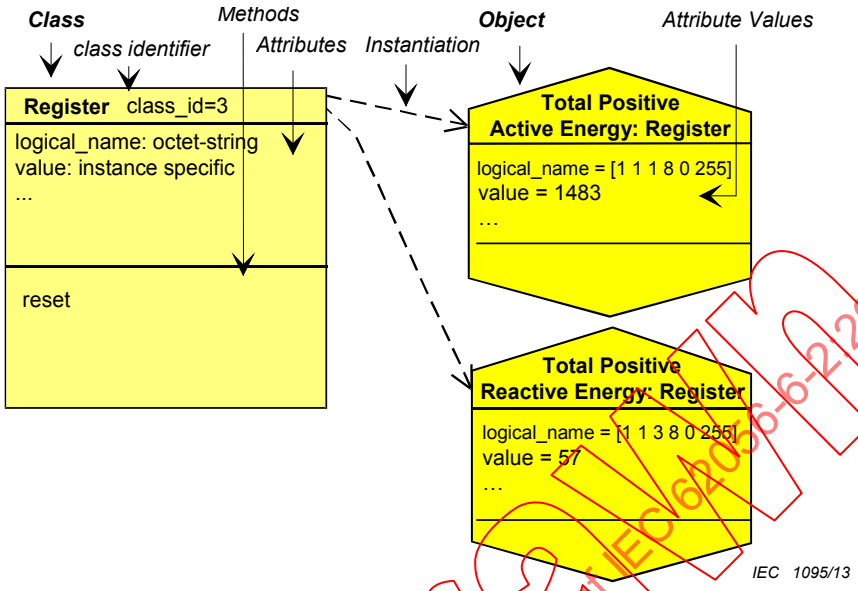
Pour les besoins de spécification, la présente norme utilise la technique de modélisation d'objets.

Un objet est un ensemble d'attributs et de méthodes. Les attributs représentent les caractéristiques d'un objet. La valeur d'un attribut peut affecter le comportement d'un objet. Le premier attribut de tout objet est le "logical_name" (c'est-à-dire nom logique). Il fait partie de l'identification de l'objet. Un objet peut proposer un certain nombre de méthodes pour examiner ou modifier les valeurs des attributs.

Les objets qui partagent des caractéristiques communes sont généralisés en une IC, identifiée par un class_id. Au sein d'une IC spécifique, les caractéristiques communes (attributs et méthodes) sont décrites une seule fois pour tous les objets. Les instanciations des IC sont appelées objets d'interfaces COSEM.

Les fabricants peuvent ajouter des méthodes et attributs propriétaires à n'importe quel objet; voir 4.2.

La Figure 1 illustre ces termes par un exemple:



Légende

Anglais	Français
Class	Classe
Class identifier	Identificateur de classe
Methods	Méthodes
Attributes	Attributs
Instantiation	Instanciation
Object	Objet
Attribute values	Valeurs d'attribut
Total positive active energy	Énergie active positive totale
Total positive reactive energy	Énergie réactive positive totale
Instance specific	Spécifique à l'instance

Figure 1 – Une classe d'interfaces et ses instances

L'IC "Register" est formé en combinant des caractéristiques nécessaires pour modéliser le comportement d'un registre générique (contenant des informations mesurées ou statiques) tel que vu du client (système de collecte de données, terminal portable). Le contenu du registre est identifié par l'attribut "logical_name". Le logical_name contient un identificateur OBIS (Voir CEI 62056-6-1). Le contenu (dynamique) réel du registre est contenu dans son attribut "value" (c'est-à-dire valeur).

Le fait de définir un compteur spécifique signifie définir plusieurs objets spécifiques. Dans l'exemple de la Figure 1, le compteur contient deux registres; à savoir, deux instances spécifiques de l'IC "Register" sont instanciées. Par le truchement de l'instanciation, un objet COSEM devient un "registre d'énergie active, positive, totale" alors que l'autre devient un "registre de puissance réactive, positive, totale".

NOTE Les objets d'interfaces COSEM (instances des IC COSEM) représentent le comportement du compteur tel que vu de "l'extérieur". Par conséquent, la modification de la valeur d'un attribut – par exemple la réinitialisation de la valeur d'un registre – est toujours déclenchée de l'extérieur. Les modifications des attributs déclenchées de l'intérieur – par exemple la mise à jour de la valeur d'un registre – ne sont pas décrites dans le présent modèle.

4.2 Méthodes de référencement

Les attributs et les méthodes des objets COSEM peuvent être référencés de deux manières différentes:

À l'aide de noms logiques (référencement par LN): Dans ce cas, les attributs et les méthodes d'un objet d'interface COSEM sont référencés via l'identificateur de l'instance d'objet COSEM à laquelle ils appartiennent.

La référence pour un attribut est: class_id, la valeur de l'attribut 'logical_name', attribute_index.

La référence pour une méthode est: class_id, la valeur de l'attribut 'logical_name', method_index.

où:

- attribute_index est utilisé comme l'identificateur de l'attribut requis. Les indices d'attribut sont spécifiés dans la définition de chaque IC. Ce sont des nombres positifs commençant à 1. Des attributs propriétaires peuvent être ajoutés: ceux-ci doivent être identifiés par des nombres négatifs;
- method_index est utilisé comme l'identificateur de la méthode requise. Les indices de méthode sont spécifiés dans la définition de chaque IC. Ce sont des nombres positifs commençant à 1. Des méthodes propriétaires peuvent être ajoutées: celles-ci doivent être identifiées par des nombres négatifs.

À l'aide de noms courts (référencement par SN): Cette sorte de référencement est destinée à être utilisée dans des dispositifs simples. Dans ce cas, chaque attribut et chaque méthode d'un objet COSEM sont identifiés par un nombre entier de 13 bits. La syntaxe pour le nom court est la même que celle du nom d'une variable nommée selon la DLMS. Le 4.4.6 de la CEI 61334-4-41:1996 et l'Article 8 de la CEI 62056-5-3:— s'appliquent.

4.3 Base_name réservés pour des objets COSEM spéciaux

Afin de faciliter l'accès à des dispositifs à l'aide du référencement par SN, certains short_name sont réservés comme des base_name pour des objets COSEM spéciaux. La plage pour les base_name réservés est comprise entre 0x**FA00** et 0x**FFF8**. Les base_name spécifiques suivants sont définis, voir Tableau 1:

Tableau 1 – Base_name réservés pour le référencement par SN

Base_name (objectName)	Objet COSEM
0xFA00	Association SN
0xFB00	Script table (instanciation: "broadcast_script_table")
0xFC00	Affectation de SAP
0xFD00	Objet "Data" (Donnée) ou "Register" (Registre) contenant le "Nom d'appareil logique COSEM" dans l'attribut "value"

4.4 Notation pour la description de classes

Le présent paragraphe 4.4 décrit la notation utilisée pour définir les IC.

Un texte court décrit la fonctionnalité et l'application de l'IC. Un tableau donne une vue d'ensemble de l'IC, y compris le nom de classe, les attributs et les méthodes. Chaque attribut et chaque méthode doivent être décrits dans le détail. Le modèle est montré ci-dessous.

Nom de la classe		Cardinalité	class_id, version			
Attribut(s)		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. …	(…)	…				x + 0x…
3. …	(…)	…				x + 0x…
Méthodes spécifiques (si requises)		***m/o***				
1.		…				x + 0x…
2.		…				x + 0x…
3.		…				x + 0x…

Nom de la classe

Décrit la classe d'interfaces (par exemple “Register”, “Clock”, “Profile generic”...)

Cardinalité

Spécifie le nombre d'instances de l'IC au sein d'un dispositif logique (Voir 4.8).

value (valeur) — L'IC doit être instanciée exactement “value” fois.

min...max. — L'IC doit être instanciée au moins “min.” fois et au plus “max.” fois. Si min. est zéro (0), l'IC est facultative, autrement (min. > 0) "min." instanciations de l'IC sont obligatoires.

class_id

Code d'identification de l'IC (plage de 0 à 65 535). Le class_id de chaque objet est récupéré avec le nom logique par lecture de l'attribut object_list d'un objet “LN d'Association” / ”SN d'Association”.

Les class_id compris entre 0 et 8 191 sont réservés pour être spécifiés par la DLMS UA.

Les class_id compris entre 8 192 et 32 767 sont réservés pour des IC spécifiques à un fabricant.

Les class_id compris entre 32 768 et 65 535 sont réservés pour des IC spécifiques à un groupe d'utilisateurs.

La DLMS UA réserve le droit d'attribuer les plages aux fabricants individuels ou aux groupes d'utilisateurs.

Version

Code d'identification de la version de l'IC. La version de chaque objet est récupérée avec le class_id et le nom logique par lecture de l'attribut object_list d'un objet “LN d'Association” / ”SN d'Association”.

Au sein d'un même dispositif logique, toutes les instances d'une certaine IC doivent être de la même version.

Attribut(s)

Spécifie l'/les attribut(s) qui appartien(nen)t à l'IC.

(*dyn.*) — Classifie un attribut qui porte une valeur traitée, qui est mis à jour par le compteur lui-même.

	(static)	Classifie un attribut, qui n'est pas mis à jour par le compteur lui-même (par exemple: données de configuration). NOTE Il existe certains attributs qui peuvent être soit statiques, soit dynamiques en fonction de l'application. Dans ces cas, cette propriété n'est pas indiquée.
logical_name	octet-string	Le “logical_name” (c'est-à-dire nom logique) est toujours le premier attribut d'une IC. Il identifie l'instanciation (objet COSEM) de cette IC. La valeur du logical_name est conforme à l'OBIS; voir Article 6 et la CEI 62056-6-1.
Type de données	Définit le type de données d'un attribut; voir 4.5.	
Min.	Spécifie si l'attribut a une valeur minimum. *X* L'attribut a une valeur minimum. <*vide*> L'attribut n'a pas de valeur minimum.	
Max.	Définit si l'attribut a une valeur maximum. *X* L'attribut a une valeur maximum. <*vide*> L'attribut n'a pas de valeur maximum.	
Def.	Spécifie si l'attribut a une valeur par défaut. Il s'agit de la valeur de l'attribut après réinitialisation. X L'attribut a une valeur par défaut. <*vide*> La valeur par défaut n'est pas définie par la définition de l'IC.	
Nom court	Lorsque le référencement par nom court (SN) est utilisé, chaque attribut et chaque méthode des instances d'un objet doivent être mis en correspondance avec des noms courts. Le base_name x de chaque instance d'objet est la variable nommée selon la DLMS à laquelle l'attribut nom logique est mis en correspondance. Il est sélectionné dans la phase de mise en œuvre. La définition de l'IC spécifie les décalages pour les autres attributs et pour les méthodes.	
Méthode(s) spécifique(s)	Fournit une liste de méthodes spécifiques qui appartiennent à l'objet. Nom de méthode () La méthode doit être décrite dans la sous-section "Description de méthode".	
m/o	Définit si la méthode est obligatoire ou facultative. *m (obligatoire)* La méthode est obligatoire. *o (facultative)* La méthode est facultative.	

Description d'attribut

Décrit chaque attribut avec son type de données (si le type de données n'est pas simple), son format de données et ses propriétés (valeurs minimum, maximum et valeurs par défaut).

Description de la méthode

Décrit chaque méthode et le comportement invoqué de l'objet ou des objets COSEM instanciés.

NOTE Les services permettant d'accéder aux attributs ou aux méthodes par le protocole sont décrits dans la CEI 62056-5-3:—, 6.6 à 6.15.

Accès sélectif

Les services xDLMS "Read", "Write", "UnconfirmedWrite" (utilisés avec le référencement par SN) et GET, SET (utilisés avec le référencement par LN) référencent typiquement l'attribut au complet. Cependant, pour certains attributs, un accès sélectif à juste une partie de l'attribut peut être assuré. La partie de l'attribut est identifiée par des paramètres d'accès sélectif spécifiques. Ceux-ci sont définis comme partie intégrante de la spécification de l'attribut.

4.5 Types de données communs

Le Tableau 2 suivant contient la liste des types de données utilisables pour les attributs des objets COSEM.

Tableau 2 – Types de données communs

Description de type	Étiquette [a]	Définition	Plage de valeurs
-- types de données simples			
null-data	[0]		
boolean	[3]	Booléen	TRUE ou FALSE
bit-string	[4]	Séquence ordonnée de valeurs booléennes	
double-long	[5]	Integer32	-2 147 483 648... 2 147 483 647
double-long-unsigned	[6]	Unsigned32	0...4 294 967 295
octet-string	[9]	Séquence ordonnée d'octets (multiple de huit bits)	
visible-string	[10]	Séquence ordonnée de caractères ASCII	
	[11]	Étiquette du type “time” dans la CEI 61334-4-41, non utilisable dans la DLMS/COSEM. Voir étiquette [27]	
UTF8-string	[12]	Séquence ordonnée de caractères codés en UTF-8	
bcd	[13]	décimal codé binaire	
integer	[15]	Integer8	-128...127
long	[16]	Integer16	-32 768...32 767
unsigned	[17]	Unsigned8	0...255
long-unsigned	[18]	Unsigned16	0...65 535
long64	[20]	Integer64	$-2^{63}...2^{63}-1$
long64-unsigned	[21]	Unsigned64	$0...2^{64}-1$
enum	[22]	Les éléments du type enumération (énumération) sont définis dans la section "Description d'attribut" dans une spécification IC COSEM.	0...255
float32	[23]	OCTET STRING (SIZE(4))	Pour le formatage, voir 4.6.2.
float64	[24]	OCTET STRING (SIZE(8))	
date_time	[25]	OCTET STRING SIZE(12))	Pour le formatage, voir 4.6.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	
-- types de données complexes			
array	[1]	Les éléments du type array (tableau) sont définis dans la section "Description d'attribut" dans une spécification IC COSEM.	
structure	[2]	Les éléments du type structure sont définis dans la section "Description d'attribut" dans une spécification IC COSEM.	
compact array	[19]	Les éléments du type compact array (tableau compact) sont définis dans la section "Description d'attribut" dans une spécification IC COSEM.	
-- CHOICE		Pour certains attributs de certains objets d'interface COSEM, le type de données peut être choisi au moment de l'instanciation de l'objet COSEM, dans la phase de mise en œuvre du serveur COSEM. Le serveur doit toujours renvoyer le type de données et la valeur de chaque attribut, ce qui, avec le nom logique, assure une interprétation non ambiguë. La liste des types de données possibles est définie dans la section "Description d'attribut" dans une spécification IC COSEM.	

a Ces étiquettes sont telles que définies dans la CEI 62056-5-3:—, Article 8.

4.6 Formats de données

4.6.1 Formats des date et heure

Les informations de date et heure peuvent être représentées avec le type de données octet-string ou à l'aide des types de données *date, time* et *date_time*, tels que définis dans la définition d'IC correspondante.

NOTE 1 Dans les futures versions des IC et dans les IC nouvellement définies, les types de données *date, time* et *date_time* seront utilisés selon le cas approprié.

NOTE 2 Les spécifications de (SIZE()) ne sont pas applicables si *date*, *time* ou *date_time* sont représentés par le type de données octet-string.

date OCTET STRING (SIZE(5))
{
year highbyte,
year lowbyte,
month,
day of month,
day of week
}
year: interprété comme long-unsigned
plage 0…big
0xFFFF = non spécifiée
year highbyte et year lowbyte référencent les deux octets du long-unsigned

month: interprété comme unsigned
plage 1…12, 0xFD, 0xFE, 0xFF
1 est janvier
0xFD = fin_changement d'heure_legale
0xFE = debut_changement_d'heure_legale
0xFF = non spécifié

dayOfMonth: interprété comme unsigned
plage 1…31, 0xFD, 0xFE, 0xFF
0xFD = avant-dernier jour du mois
0xFE = dernier jour du mois
0xE0 à 0xFC = réservés
0xFF = non spécifié

dayOfWeek: interprété comme unsigned
plage 1…7, 0xFF
1 est lundi
0xFF = non spécifié

Pour dayOfMonth et dayOfWeek:

Pour les dates répétitives, les parties non utilisées doivent être mises à "non spécifiée"

Les éléments dayOfMonth et dayOfWeek doivent être interprétés ensemble.

- Si le dernier dayOfMonth est spécifié (0xFE) et dayOfWeek est un joker, cela spécifie le dernier jour calendaire du mois;
- Si le dernier dayOfMonth est spécifié (0xFE) et un dayOfWeek explicite est spécifié (par exemple 7, dimanche), il s'agit de la dernière occurrence du weekday (jour de la semaine) spécifié dans le month (mois), c'est-à-dire le dernier dimanche.
- Si year (l'année) n'est pas spécifié (FFFF), et dayOfMonth et dayOfWeek sont tous deux explicitement spécifiés, il doit être interprété comme le dayOfWeek

du dayOfMonth ou suivant celui-ci.

- Si year (l'année) et month (le mois) sont spécifiés et dayOfMonth et dayOfWeek sont tous deux explicitement spécifiés mais les valeurs ne sont pas compatibles, cela doit être considéré comme étant une erreur.

EXEMPLE 1 year = 0xFFFF, month = FF, dayOfMonth = 0xFE, dayofWeek = 0xFF: le dernier jour du mois dans chaque année et chaque mois;

EXEMPLE 2 year = 0xFFFF, month = FF, dayOfMonth = 0xFE, dayofWeek = 0x07: le dernier dimanche de chaque année et chaque mois;

EXEMPLE 3 year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayofWeek = 0x07: le dernier dimanche de mars de chaque année;

EXEMPLE 4 year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayofWeek = 0x07: le premier dimanche de mars de chaque année;

EXEMPLE 5 year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayofWeek = 0x05: le quatrième vendredi de mars de chaque année;

EXEMPLE 6 year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayofWeek = 0x07: le quatrième dimanche d'octobre de chaque année;

EXEMPLE 7 year = 0x07D9, month = 0x01, dayOfMonth = 0x13, dayofWeek = 0x01: 2009.01.19, lundi;

EXEMPLE 8 year = 0x07D9, month = 0x01, dayOfMonth = 0x13, dayofWeek = 0x02: erreur, car le dayOfMonth et le dayOfWeek dans l'année et le mois donnés ne concordent pas.

time OCTET STRING (SIZE(4))
{
hour,
minute,
second,
hundredths
}
hour: interprété comme unsigned
plage 0…23, 0xFF
minute: interprété comme unsigned
plage 0…59, 0xFF
second: interprété comme unsigned
plage 0…59, 0xFF,
hundredths: interprété comme unsigned
plage 0…99, 0xFF

Pour hour (c'est-à-dire heure), minute, second (c'est-à-dire seconde) et hundredths (c'est-à-dire centièmes): 0xFF = non spécifié.

Pour les heures répétitives, les parties non utilisées doivent être mises à "non spécifiée".

deviation long –720…720:
en minutes de l'heure locale par rapport à l'heure GMT
0x8000 = non spécifié

clock_status unsigned interprété comme une chaîne de huit bits
Les bits de statut sont définis comme suit:
bit 0 (LSB): valeur non valide [a],
bit 1: valeur douteuse [b],
bit 2: base d'horloge différente [c],
bit 3: statut d'horloge non valide [d],
bit 4: réservé,
bit 5: réservé,
bit 6: réservé,
bit 7 (MSB): heure d'été/hiver active [e]

date_time OCTET STRING (SIZE(12))

{
year highbyte,
year lowbyte,
month,
day of month,
day of week,
hour,
minute,
second,
hundredths of second,
deviation highbyte,
deviation lowbyte,
clock status
}

les champs individuels de *date_time* sont codés comme définis ci-dessus. Certains peuvent être mis à “non spécifié“ comme décrit ci-dessus dans *date* et *time*.

[a] L'heure ne pouvait pas être récupérée après un incident. Les conditions détaillées sont spécifiques au fabricant (par exemple après une coupure de l'alimentation de l'horloge).

[b] L'heure pouvait être récupérée après un incident mais la valeur ne peut pas être garantie. Les conditions détaillées sont spécifiques au fabricant.

[c] Le bit est positionné si les informations de temps élémentaires pour l'horloge à l'instant effectif proviennent d'une source de temps différente de la source spécifiée dans clock_base.

[d] Ce bit indique qu'au moins un bit du statut de l'horloge est non valide. Certains bits peuvent être corrects. La signification exacte doit être expliquée dans la documentation du fabricant.

[e] Fanion mis à vrai: l'heure transmise contient l'écart été/hiver (heure d'été).

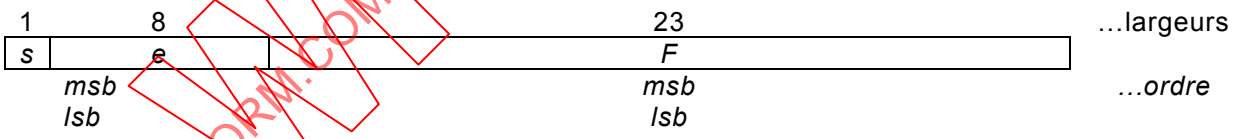
Fanion mis à faux: l'heure transmise ne contient pas l'écart été/hiver (heure normale).

4.6.2 Formats de nombres en virgule flottante

Les formats de nombres en virgule flottante sont définis dans l'ISO/CEI/IEEE 60559.

NOTE Pour ce qui suit, l'ISO/CEI/IEEE 60559 équivaut à l'IEEE 754.

Le format "single" (simple) est:



où:

– *s* est le bit de signe;
– *e* est l'exposant; il a une largeur de 8 bits et le biais d'exposant est +127;
– *f* est la fraction, il est de 23 bits.

Avec ces valeurs, la valeur est (si 0 < e < 255):

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

Le format "double" (double) est:

1	11	52	...largeurs
s	*e*	*F*	
msb *lsb*		*msb* *lsb*	*...ordre*

où:

– *s* est le bit de signe;
– *e* est l'exposant; il a une largeur de 11 bits et le biais d'exposant est +1 023;
– *f* est la fraction, il est de 52 bits.

Avec ces valeurs, la valeur est (si 0 < e < 2 047):

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

Pour plus d'informations, voir l'ISO/CEI/IEEE 60559.

Les nombres en virgule flottante doivent être représentés sous la forme d'une chaîne d'octets de longueur fixe, contenant les quatre octets (float32) du nombre en virgule flottante de format single ou les huit octets (float64) du nombre en virgule flottante de format double tels que spécifiés ci-dessus, l'octet de poids fort en premier.

EXEMPLE 1 La valeur décimale "1" représentée au format virgule flottante single est:

Bit 31 Bit de signe **0** 0: + 1: -	Bits 30 à 23: Champ exposant: **01111111** La valeur décimale du champ exposant et l'exposant: **127**-127 = 0	Bits 22 à 0: Significande 1,**00000000000000000000000** Valeur décimale du significande: 1,0000000

NOTE Le significande est le nombre binaire 1 suivi de la séparation fractionnaire suivie des bits binaires de la fraction.

Le codage, y compris l'étiquette du type de données est (toutes les valeurs sont hexadécimales): 17 3F 80 00 00.

EXEMPLE 2 La valeur décimale "1" représentée au format virgule flottante double est:

Bit 63 Bit de signe **0** 0: + 1: -	Bits 62 à 52: Champ exposant: **01111111111** La valeur décimale du champ exposant et l'exposant: **1023**-1023 = 0	Bits 51 à 0: Significande 1,**0000000000000000000000000000000000000000000000000000** Valeur décimale du significande: 1,0000000000000000

Le codage, y compris l'étiquette du type de données est (toutes les valeurs sont hexadécimales):
18 3F F0 00 00 00 00 00.

EXEMPLE 3 La valeur décimale "62056" représentée au format virgule flottante single est:

Bit 31 Bit de	Bits 30 à 23: Champ exposant: **10001110**	Bits 22 à 0: Significande

signe **0** 0: + 1: -	La valeur décimale du champ exposant et l'exposant: **142**-127 = 15	1,**11100100110100000000000** Valeur décimale du significande: 1,8937988

Le codage, y compris l'étiquette du type de données est (toutes les valeurs sont hexadécimales): 17 47 72 68 00.

EXEMPLE 4 La valeur décimale “62056” représentée au format virgule flottante double est:

Bit 63 Bit de signe **0** 0: + 1: -	Bits 62 à 52: Champ exposant: **10000001110** La valeur décimale du champ exposant et l'exposant: **1038**-1023 = 15	Bits 51 à 0: Significande 1,**11100100110100000000000000000000000000000000000000000** Valeur décimale du significande: 1,8937988281250000

Le codage, y compris l'étiquette du type de données est (toutes les valeurs sont hexadécimales): 18 40 EE 4D 00 00 00 00 00.

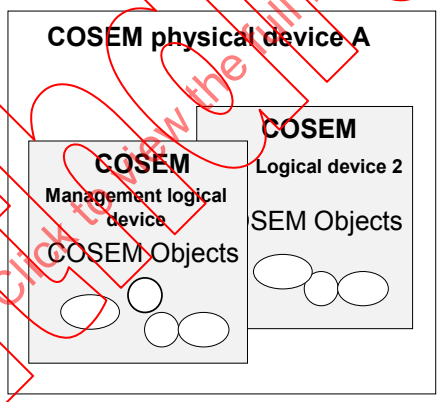
4.7 Le modèle de serveur COSEM

Le serveur COSEM est structuré en trois niveaux hiérarchiques comme montré à la Figure 2:

Niveau 1: Dispositif physique,

Niveau 2: Dispositif logique;

Niveau 3: Objets COSEM accessibles.

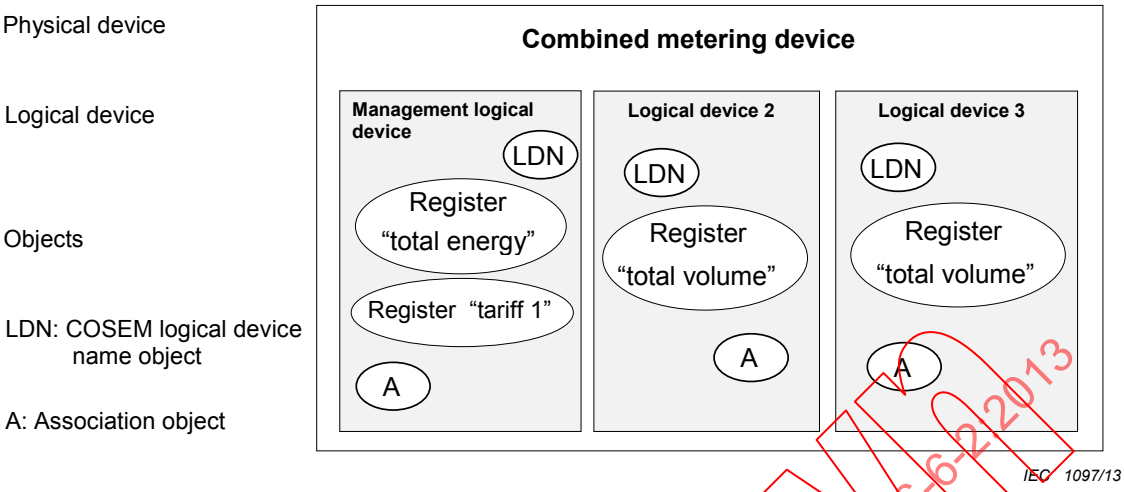


Légende

Anglais	Français
COSEM physical device A	Dispositif physique COSEM A
Management logical device	Dispositif logique de gestion
COSEM Objects	Objets COSEM

Figure 2 – Le modèle de serveur COSEM

L'exemple suivant dans la Figure 3 montre comment un dispositif de comptage combiné peut être structuré à l'aide du modèle de serveur COSEM.



Légende

Anglais	Français
Physical device	Dispositif physique
Logical device	Dispositif logique
Combined metering device	Dispositif de comptage combiné
Management logical device	Dispositif logique de gestion
Logical device	Dispositif logique
Objects	Objets
LDN: COSEM logical device name object	LDN: objet nom de dispositif logique COSEM
A: Association object	A: Objet Association
Register « total energy »	Registre « énergie totale »
Register « total volume »	Registre « volume total"

Figure 3 – Dispositif de comptage combiné

4.8 Le dispositif logique COSEM

4.8.1 Généralités

Le dispositif logique COSEM contient un ensemble d'objets COSEM. Chaque dispositif physique doit contenir un "Dispositif logique de gestion".

L'adressage des dispositifs logiques COSEM doit être assuré par le plan d'adressage des couches inférieures de la pile protocolaire utilisée.

4.8.2 Nom de dispositif logique COSEM

Le dispositif logique COSEM peut être identifié par son propre nom de dispositif logique COSEM. Ce nom peut être récupéré à partir d'une instance "d'affectation de SAP"" d'IC (voir 5.3.3) ou à partir d'un objet COSEM "Nom de dispositif logique COSEM" (voir 6.2.24).

Le nom de dispositif logique est défini comme étant une chaîne d'octets de 16 octets au maximum. Les trois premiers octets doivent porter l'identificateur du fabricant[5]. Le fabricant doit assurer que le nom de dispositif logique, commençant par les trois octets identifiant le fabricant et suivi de 13 octets au maximum, est unique.

5 Administré par la DLMS User Association (Association des utilisateurs de la DLMS).

4.8.3 La "vue association" du dispositif logique

Afin d'accéder aux objets COSEM dans le serveur, une association d'applications (AA) doit d'abord être établie avec un client. Celle-ci identifie les partenaires et caractérise le contexte dans lequel les applications associées communiqueront. Les principales parties de ce contexte sont:

- le contexte d'application;
- le contexte d'authentification;
- le contexte xDLMS.

L'AA est modélisée par des objets COSEM spéciaux: les objets "Association". Il est défini deux types de ces objets:

- un pour l'utilisation du référencement par SN ("SN d'association", voir 5.3.1);
- et l'un pour l'utilisation du référencement par LN ("LN d'association", 5.3.2).

En fonction de l'AA établie entre le client et le serveur, des droits d'accès différents peuvent être accordés par le serveur. Les droits d'accès concernent un ensemble d'objets COSEM – les objets visibles – qui peuvent être accessibles ('vus') au sein d'une AA donnée. En outre, l'accès aux attributs et méthodes de ces objets COSEM peut aussi être limité au sein de l'AA (par exemple, un certain type de client ne peut lire qu'un attribut particulier d'un objet COSEM, mais ne peut pas l'écrire).

La liste des objets COSEM visibles – la "vue association" – peut être obtenue par le client par lecture de l'attribut "*object_list*" de l'objet d'association approprié.

4.8.4 Contenu obligatoire d'un dispositif logique COSEM

Les objets suivant doivent être présents dans chaque dispositif logique COSEM. Ils doivent être accessibles pour GET/Read dans toutes les AA avec ce dispositif logique:

- Objet nom de dispositif logique;
- objet association actuelle (LN ou SN).

NOTE Si l'objet Affectation de SAP est présent, l'objet Nom de dispositif logique COSEM peut ne pas être présent.

4.8.5 Dispositif logique de gestion

Comme spécifié en 4.8.1, le dispositif logique de gestion est un élément obligatoire de tout dispositif physique. Il a une adresse réservée. Il doit prendre en charge une AA avec un client public avec le niveau de sécurité le plus bas. Son rôle est de prendre en charge la révélation de la structure interne du dispositif physique et de prendre en charge la notification d'événements dans le serveur.

En plus de l'objet "Association" modélisant l'AA avec le client public, le dispositif logique de gestion doit contenir un objet "Affectation de SAP", en donnant son SAP et le SAP de tous les autres dispositifs logiques au sein du dispositif physique. L'objet Affectation de SAP doit être lisible au moins par le client public.

S'il n'y a qu'un seul dispositif logique au sein du dispositif physique, l'objet "Affectation de SAP" peut être omis.

4.9 Sécurité des données

DLMS/COSEM fournit plusieurs fonctions de sécurité des informations pour l'accès et le transport des données:

- la *sécurité d'accès des données* contrôle l'accès aux données maintenues par un serveur DLMS/COSEM;

- la *sécurité de transport des données* permet à la partie émittrice d'appliquer une protection chiffrée aux APDU de xDLMS envoyées. Cela exige des APDU chiffrées. La partie réceptrice peut retirer ou vérifier cette protection.

Pour une description de ces mécanismes de sécurité, voir la CEI 62056-5-3:—, Article 5.

La sécurité des données est fournie par la couche Application de la COSEM et elle et prise en charge/gérée par les objets suivants:

- SN d'association, voir 5.3.1;
- LN d'association, voir 5.3.2; et
- Établissement de la sécurité, voir 5.3.5.

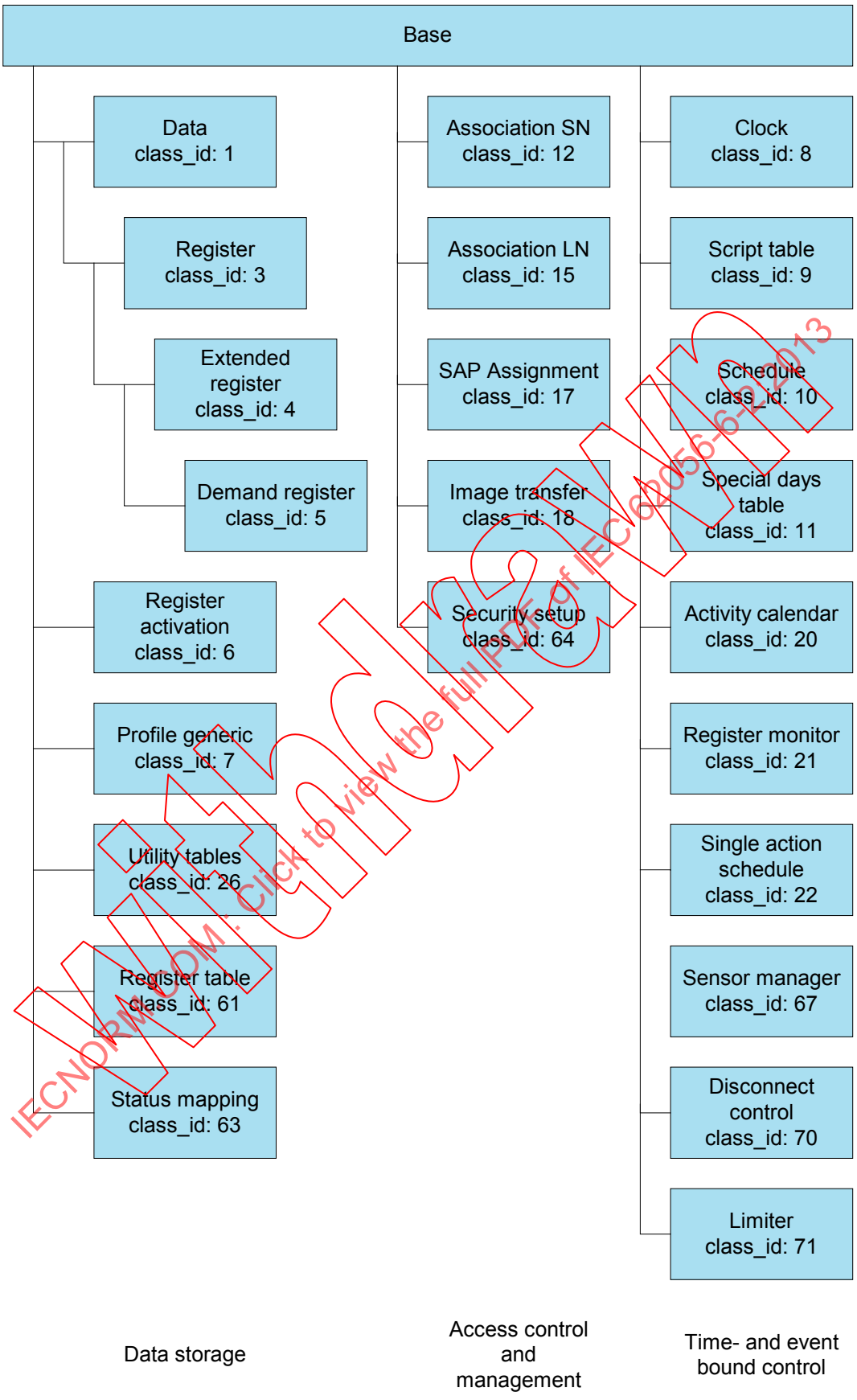
5 Les classes d'interfaces de la COSEM

5.1 Vue d'ensemble

Les classes d'interfaces définies à l'heure actuelle et les relations entre elles sont montrées à la Figure 4 et à la Figure 5.

NOTE 1 La “base” d'IC proprement dite n'est pas spécifiée de façon explicite. Elle contient un seul attribut "logical_name".

NOTE 2 Dans la description des IC "Demand register" (registre de puissance), “Clock” (horloge) et “Profile generic” (profil générique), les 2nd attributs sont étiquetés différemment de celui du 2nd attribut de l'IC “Data” (données) IC, à savoir "current_average_value", “time” et “buffer” contre “value”. Le but est de mettre l'accent sur la nature spécifique de la “value”.

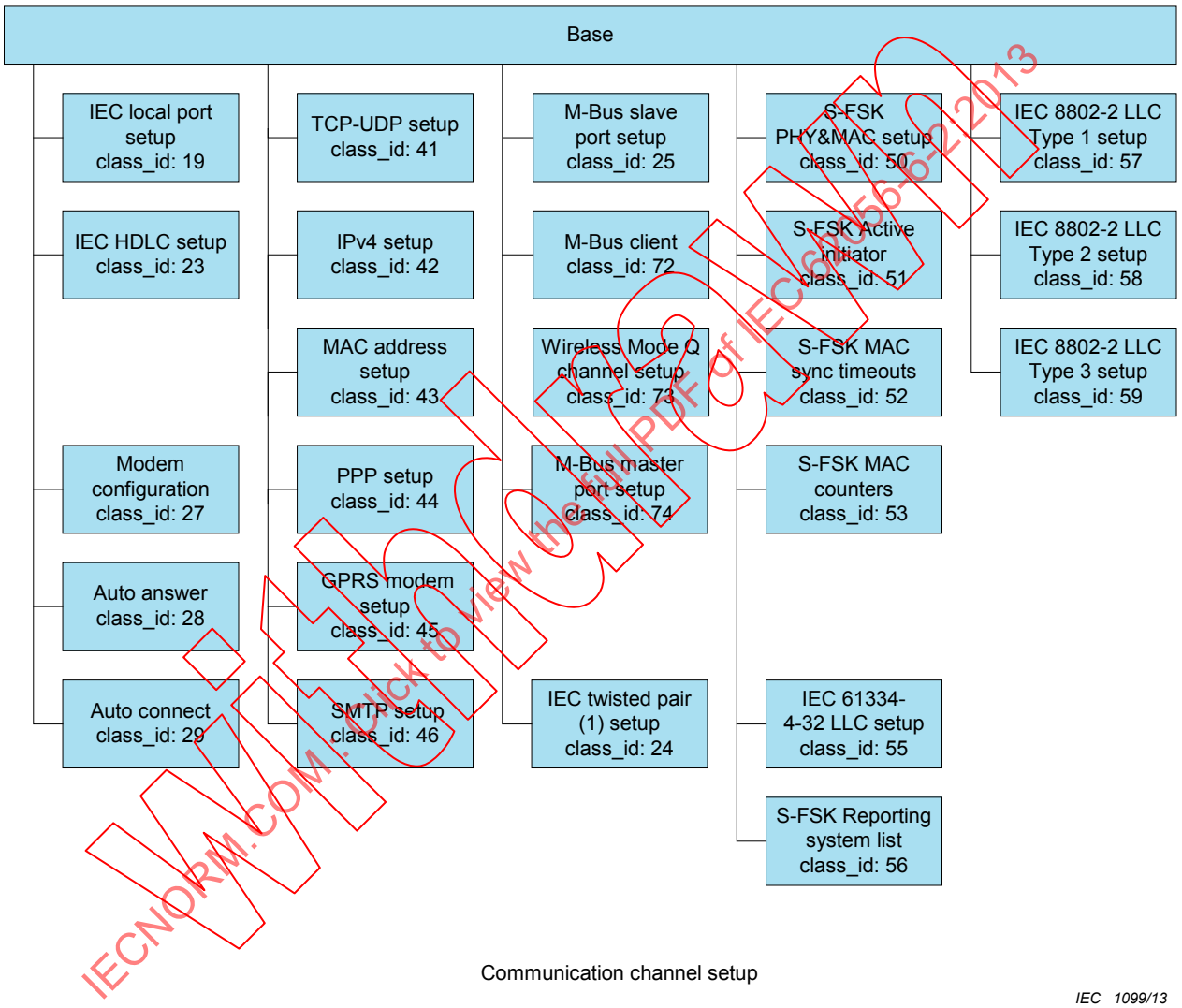


IEC 1098/13

Légende

Anglais	Français
Data storage	Stockage de données
Access control and management	Contrôle d'accès et gestion
Time and event bound control	Commande à limite temporelle et événementielle

Figure 4 – Vue d'ensemble des classes d'interfaces – Partie 1



Légende

Anglais	Français
Communication channel setup	Établissement de voie de communication

Figure 5 – Vue d'ensemble des classes d'interfaces – Partie 2

5.2 Classes d'interfaces pour paramètres et données de mesure

5.2.1 Data (class_id: 1, version: 0)

Cette IC permet de modéliser diverses données, telles que données de configuration et paramètres. Les données sont identifiées par l'attribut *logical_name*.

Données	**0...n**	**class_id = 1, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				X
2. value	*CHOICE*				X + 0x08
Méthodes spécifiques	**m/o**				

Description d'attribut

logical_name Identifie l'instance de l'objet “Data”. Voir l'Article 6 et la CEI 62056-6-1.

value Contient les données.

```
CHOICE
{
    -- types de données simples
    null-data                [0],
    boolean                  [3],
    bit-string               [4],
    double-long              [5],
    double-long-unsigned     [6],
    octet-string             [9],
    visible-string           [10],
    UTF8-string              [12],
    bcd                      [13],
    integer                  [15],
    long                     [16],
    unsigned                 [17],
    long-unsigned            [18],
    long64                   [20],
    long64-unsigned          [21],
    enum                     [22],
    float32                  [23],
    float64                  [24],
    date-time                [25],
    date                     [26],
    time                     [27],
    -- types de données complexes
    array                    [1],
    structure                [2],
    compact-array            [19]
}
```

Le type de données dépend de l'instanciation définie par le "nom logique"" et éventuellement provenant du fabricant. Il doit être choisi de sorte à rendre possible, avec le nom logique, une interprétation non ambiguë. Tout type de données simple ou complexe énuméré en 4.5 peut être utilisé, sauf en cas de limitation du choix par l'Article 6.

5.2.2 Registre (class_id: 3, version: 0)

Cette IC permet la modélisation d’une valeur issue d’un processus ou d’un état avec le scalaire et l'unité. Les objets “Registre” ont une notion de la nature de la valeur issue du processus ou de l’état. Elle est identifiée par l'attribut *logical_name*.

Registre		0...n	class_id = 3, version = 0			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				X
2. value	(dyn.)	CHOICE				x + 0x08
3. scaler_unit	(static)	scal_unit_type				x + 0x10
Méthodes spécifiques		**m/o**				
1. reset (data)		o				x + 0x28

Description d'attribut

logical_name Identifie l'instance de l'objet “Register” (Registre). Voir l'Article 6 et la CEI 62056-6-1.

value Contient la valeur traitée ou d'état courant.

```
CHOICE
{
        -- types de données simples
        null-data                [0],
        bit-string               [4],
        double-long              [5],
        double-long-unsigned     [6],
        octet-string             [9],
        visible-string           [10],
        UTF8-string              [12],
        integer                  [15],
        long                     [16],
        unsigned                 [17],
        long-unsigned            [18],
        long64                   [20],
        long64-unsigned          [21],
        float32                  [23],
        float64                  [24]
}
```

Le type de données de la valeur dépend de l'instanciation définie par le “logical_name” et éventuellement du choix du fabricant. Il doit être choisi de sorte à rendre possible, avec le nom logique, une interprétation non ambiguë de la valeur.

Lorsqu'en lieu et place d'un objet “Data”, un objet “Register” est utilisé, (sans utilisation de l'attribut scaler_unit ou avec scalaire (échelle de comptage) = 0, unit (unité) = 255), les types de données autorisés pour l'attribut "value" de l'IC “Data” sont autorisés.

scaler_unit Fournit des informations relatives à l'unité et scalaire de la valeur.

```
scal_unit_type::= structure
{
scaler,
unit
}
```

scaler: integer

Il s'agit de l'exposant (en base 10) du facteur de multiplication.

REMARQUE Si la valeur n'est pas numérique, scaler (l'échelle de comptage) doit être mis à 0.

unit: enum

Énumeration définissant l'unité physique; pour les détails, voir le Tableau 3 ci-dessous.

Description de la méthode

reset (data) Cette méthode force une réinitialisation de l'objet. En invoquant cette

méthode, la valeur est mise à la valeur par défaut. La valeur par défaut est une constante spécifique à l'instance.

data::= integer(0)

Tableau 3 – Valeurs énumérées pour les unités physiques

unit:: = enum	Unit	Grandeur	Nom de l'unité	Définition SI (commentaire)
(1)	a	temps	year (c'est-à-dire année)	
(2)	mo	temps	month (c'est-à-dire mois)	
(3)	wk	temps	week (c'est-à-dire semaine)	7*24*60*60 s
(4)	d	temps	day (c'est-à-dire jour)	24*60*60 s
(5)	h	temps	heure	60*60 s
(6)	min.	temps	min	60 s
(7)	s	temps (t)	seconde	s
(8)	°	(phase) angle	degré	rad*180/π
(9)	°C	température (T)	degré-celsius	K-273.15
(10)	currency	monnaie (locale)		
(11)	m	longueur (l)	mètre	M
(12)	m/s	vitesse (v)	mètre par seconde	m/s
(13)	m^3	volume (V) $r_{V,}$ constante du compteur ou valeur d'impulsion (volume)	mètre cube	m^3
(14)	m^3	volume corrigé	mètre cube	m^3
(15)	m^3/h	débit volumique	mètre cube par heure	$m^3/(60*60s)$
(16)	m^3/h	débit volumique corrigé	mètre cube par heure	$m^3/(60*60s)$
(17)	m^3/d	débit volumique		$m^3/(24*60*60s)$
(18)	m^3/d	débit volumique corrigé		$m^3/(24*60*60s)$
(19)	l	volume	litre	10^{-3} m^3
(20)	kg	masse (m)	kilogramme	
(21)	N	force (F)	newton	
(22)	Nm	énergie	newton-mètre	J = Nm = Ws
(23)	Pa	pression (p)	pascal	N/m^2
(24)	bar	pression (p)	bar	10^5 N/m^2
(25)	J	énergie	joule	J = Nm = Ws
(26)	J/h	puissance thermique	joule par heure	J/(60*60s)
(27)	W	puissance active (P)	watt	W = J/s
(28)	VA	Puissance apparente (S)	voltampère	
(29)	var	puissance réactive (Q)	var	
(30)	Wh	énergie active $r_{W,}$ constante d'énergie active du compteur ou valeur d'impulsion	wattheure	W*(60*60s)
(31)	VAh	énergie apparente $r_{S,}$ constante d'énergie apparente du compteur ou valeur d'impulsion	volt ampèreheure	VA*(60*60s)
(32)	varh	énergie réactive $r_{B,}$ constante d'énergie réactive du compteur ou valeur d'impulsion	varheure	var*(60*60s)

unit:: = enum	Unit	Grandeur	Nom de l'unité	Définition SI (commentaire)
(33)	A	courant (I)	ampère	A
(34)	C	charge électrique (Q)	coulomb	C = As
(35)	V	tension (U)	volt	V
(36)	V/m	intensité du champ électrique (E)	volt par mètre	V/m
(37)	F	capacité (C)	farad	C/V = As/V
(38)	Ω	résistance (R)	ohm	Ω = V/A
(39)	$\Omega m^2/m$	résistivité (ρ)		Ωm
(40)	Wb	flux magnétique (Φ)	weber	Wb = Vs
(41)	T	induction magnétique (B)	tesla	Wb/m^2
(42)	A/m	intensité du champ magnétique (H)	ampère par mètre	A/m
(43)	H	inductance (L)	henry	H = Wb/A
(44)	Hz	fréquence (f, ω)	hertz	1/s
(45)	1/(Wh)	R_W, constante d'énergie active du compteur ou valeur d'impulsion		
(46)	1/(varh)	R_B, constante d'énergie réactive du compteur ou valeur d'impulsion		
(47)	1/(VAh)	R_S, constante d'énergie apparente du compteur ou valeur d'impulsion		
(48)	V^2h	volt carré-heure r_{U2h}, constante volt carré-heure du compteur ou valeur d'impulsion	volt carré-heures	$V^2(60*60s)$
(49)	A^2h	ampère carré-heure r_{I2h}, constante ampère carré heure du compteur ou valeur d'impulsion	ampère carré-heures	$A^2(60*60s)$
(50)	kg/s	débit massique	kilogramme par seconde	kg/s
(51)	S, mho	conductance	siemens	1/Ω
(52)	K	température (T)	kelvin	
(53)	$1/(V^2h)$	R_{U2h}, constante volt carré heure du compteur ou valeur d'impulsion		
(54)	$1/(A^2h)$	R_{I2h}, constante ampère carré heure du compteur ou valeur d'impulsion		
(55)	$1/m^3$	R_V, constante du compteur ou valeur d'impulsion (volume)		
(56)		pourcentage	%	
(57)	Ah	ampères heures	Ampèreheure	
…				
(60)	Wh/m^3	énergie volumique	$3,6*10^3$ J/m^3	
(61)	J/m^3	valeur calorifique, wobbe		
(62)	Mol %	fraction molaire de la composition gazeuse	mole pourcent	(Unité élémentaire de composition gazeuse)
(63)	g/m^3	masse volumique, quantité de matière		(Analyse gazeuse, éléments d'accompagnement)
(64)	Pa s	viscosité dynamique	pascal seconde	(Caractéristique d'un flux gazeux)
(65)	J/kg	énergie spécifique NOTE La quantité d'énergie par unité de masse d'une substance	Joule par kilogramme	$m^2 \cdot kg \cdot s^{-2} / kg = m^2 \cdot s^{-2}$
….				

unit:: = enum	Unit	Grandeur	Nom de l'unité	Définition SI (commentaire)
(70)	dBm	Intensité de signal (par exemple du système radio GSM)		
…				
(253)		reserved (réservé)		
(254)	other (autre)	autre unité		
(255)	count (compte)	sans unité, sans dimension, compte		

Un certain nombre d'exemples sont montrés au Tableau 4 ci-dessous.

Tableau 4 – Exemples pour scaler_unit

Valeur	Scalaire	Unité	Données
263788	-3	m^3	263,788 m^3
593	3	Wh	593 kWh
3467	-1	V	346,7
3467	0	V	3 467 V
3467	1	V	34 670

5.2.3 Registre étendu (class_id: 4, version: 0)

Cette IC permet de modéliser une valeur issue d'un processus avec ses informations associées relatives au scalaire, à l'unité, à l'état et à l'instant d'acquisition. Les objets " Extended register" (Registre étendu) ont une notion de la nature de la valeur issue du processus. Elle est décrite par l'attribut *logical_name*.

Extended register (registre étendu)		**0…n**	**class_id = 4, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				X
2. value	(dyn.)	CHOICE				x + 0x08
3. scaler_unit	(static)	scal_unit_type				x + 0x10
4. status	(dyn.)	CHOICE				x + 0x18
5. capture_time	(dyn.)	octet-string				x + 0x20
Méthodes spécifiques		***m/o***				
1. reset (data)		o				x + 0x38

Description d'attribut

Pour la définition des attributs *value* et *scaler_unit*, voir la description de l'IC "Register".

logical_name Identifie l'instance de l'objet “Extended register” (Registre étendu). Voir 6.3.1.

status Fournit des informations d'état spécifiques à “Extended register”. La signification des éléments d'état doit être fournie pour chaque instance de l'objet “Extended register”.

CHOICE
{

Le type de données et le codage dépendent de l'instanciation et éventuellement du choix du fabricant.

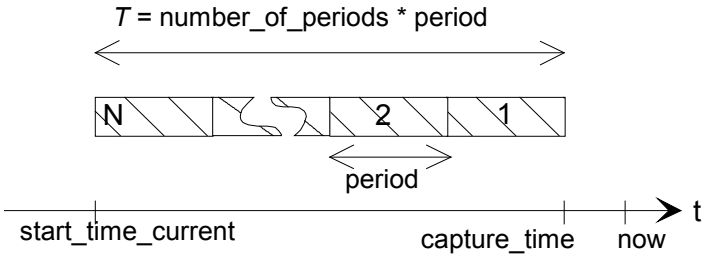
	-- types de données simples null-data [0], bit-string [4], double-long-unsigned [6], octet-string [9], visible-string [10], UTF8-string [12], unsigned [17], long-unsigned [18], long64-unsigned [21] }	Pour l'interprétation, des informations complémentaires fournies par le fabricant peuvent être nécessaires.
	Def.	Dépendant de la définition du type de statut.
capture_time	Fournit les informations de date et d'heure spécifiques au registre étendu "Extended register" montrant le moment où la valeur de l'attribut "value" a été saisie. octet-string, formaté comme indiqué en 4.6.1 pour *date_time*	

Description de la méthode

reset (data)	Cette méthode force une réinitialisation de l'objet. En invoquant cette méthode, l'attribut value est mis à la valeur par défaut. La valeur par défaut est une constante spécifique à l'instance. • L'attribut capture_time est mis à l'heure de l'exécution de la réinitialisation. data::= integer(0)

5.2.4 Registre de demande (class_id: 5, version: 0)

Cette IC permet de modéliser une valeur de puissance avec ses informations associées relatives au scalaire, aux unités, à l'état et aux instants d'acquisition. Un objet "Demand register" mesure et calcule périodiquement une valeur moyenne courante (*current_average_value*) et stocke une dernière valeur moyenne (*last_average_value*). L'intervalle de temps *T* sur lequel la demande est mesurée ou calculée est défini en spécifiant "*number_of_periods*" (le nombre de périodes) et "*period*" (période). Voir Figure 6.



T is the time interval used for the calculation of the current_value of a sliding demand register

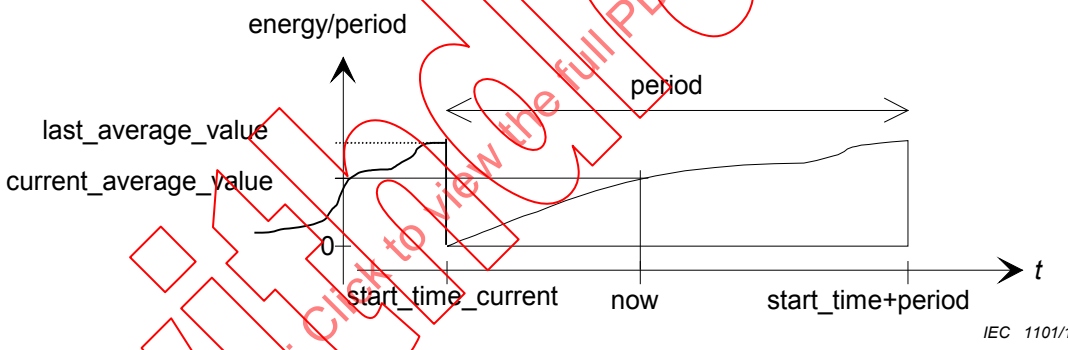
Légende

Anglais	Français
T is.... demand register	T est l'intervalle de temps utilisé pour le calcul de la current_value d'un registre de demande glissante

Figure 6 – Les attributs de temps pour mesurer une demande glissante

Le registre de puissance délivre deux types de puissance: *current_average_value* et *last_average_value* (voir Figure 7 et Figure 8).

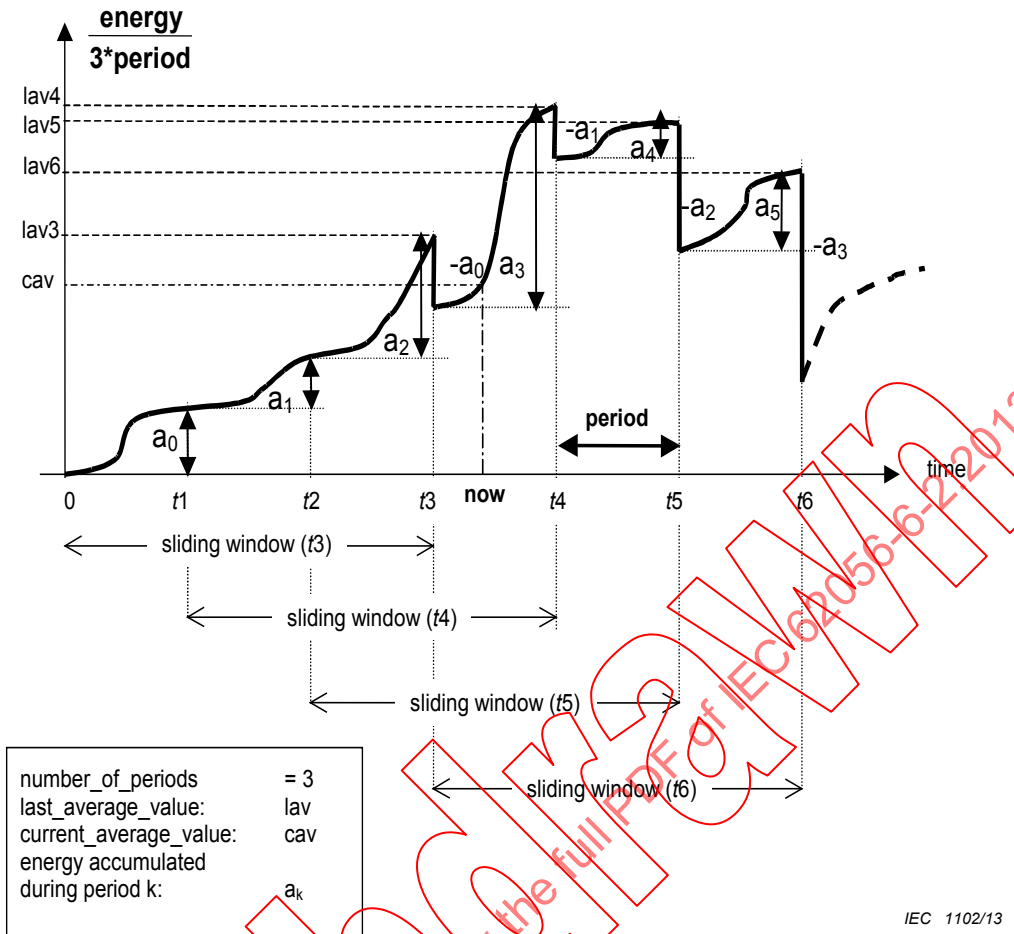
Les objets "Demand register" ont une notion de la nature de la valeur issue du processus qui est décrite par l'attribut logical_name.



Légende

Anglais	Français
energy/period	Énergie/période
period	période
now	maintenant

Figure 7 – Les attributs dans le cas de la demande en bloc



Légende

Anglais	Français
energy/period	énergie/période
period	période
now	maintenant
sliding window	fenêtre glissante
energy accumulated during period k	énergie accumulée au cours de la période k

Figure 8 – Les attributs dans le cas de la puissance glissante (nombre de périodes = 3)

Demand register (Registre de demande)		0...n	class_id = 5, version = 0			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. current_average_value	(dyn.)	CHOICE			0	x + 0x08
3. last_average_value	(dyn.)	CHOICE			0	x + 0x10
4. scaler_unit	(static)	scal_unit_type				x + 0x18
5. status	(dyn.)	CHOICE				x + 0x20
6. capture_time	(dyn.)	octet-string				x + 0x28
7. start_time_current	(dyn.)	octet-string				x + 0x30
8. period	(static)	double-long-unsigned	1			x + 0x38
9. number_of_periods	(static)	long-unsigned	1		1	x + 0x40
Méthodes spécifiques		***m/o***				
1. reset (data)		o				x + 0x48
2. next_period (data)		o				x + 0x50

Description d'attribut

Pour l'attribut *scaler_unit*, voir la description de l'IC "Register".

logical_name Identifie l'instance de l'objet "Demand register" (Registre de demandes). Voir 6.3.1.

current_average_value Fournit la valeur courante (puissance en cours) de l'énergie accumulée depuis start_time, divisée par number_of_periods*period.

Le type de données de la valeur dépend de l'instanciation définie par le "logical_name" et éventuellement du choix du fabricant. Le type doit être choisi de sorte à rendre possible, avec le nom logique, une interprétation non ambiguë de la valeur.

Si une grandeur autre que l'énergie est mesurée, d'autres méthodes de calcul peuvent s'appliquer (par exemple pour calculer les valeurs moyennes de la tension ou du courant).

CHOICE Pour les types de données, voir l'IC "Register", attribut "value".

last_average_value Fournit la valeur de l'énergie accumulée (sur les dernières number_of_periods*period) divisée par number_of_periods*period. L'énergie de la période en cours (non achevée) n'est pas prise en compte dans le calcul.

Si une grandeur autre que l'énergie est mesurée, d'autres méthodes de calcul peuvent s'appliquer (par exemple pour calculer les valeurs moyennes de la tension ou du courant).

CHOICE Pour les types de données, voir l'IC "Register", attribut "value".

status Fournit des informations d'état spécifiques à "Demand register". Le type de données et le codage dépendent de l'instanciation et éventuellement du choix du fabricant. Pour l'interprétation, des informations

	complémentaires fournies par le fabricant peuvent être nécessaires. CHOICE Pour les types de données, voir l'IC “Extended register”, attribut "status". ***Def.*** Dépendant de la définition du type d'état.
capture_time	Fournit la date et l'heure du calcul de last_average_value. octet-string, formaté comme indiqué en 4.6.1 pour *date_time*
start_time_current	Fournit la date et l'heure du début de la mesure de current_average_value. octet-string, formaté comme indiqué en 4.6.1 pour *date_time*
period	"Period" est l'intervalle entre deux mises à jour successives de last_average_value. (number_of_periods*period est le dénominateur pour le calcul de la demande). double-long-unsigned Période de mesure en secondes Le comportement du compteur après écriture d'une nouvelle valeur dans cet attribut doit être spécifié par le fabricant.
number_of_ periods	Le nombre de périodes utilisées pour calculer last_average_value. Le number_of_periods >= 1 number_of_periods > 1 indique que last_average_value représente une "demande glissante". number_of_periods = 1 indique que last_average_value représente une "demande en bloc". Le comportement du compteur après écriture d'une nouvelle valeur dans cet attribut doit être spécifié par le fabricant.
Description de la méthode	
reset (data)	Cette méthode force une réinitialisation de l'objet. L'activation de cette méthode provoque les actions suivantes: • il est mis fin à la période courante; • les valeurs current_average_value et last_average_value sont mises à leurs valeurs par défaut; • capture_time et start_time_current sont mis à l'heure de l'exécution de *reset (data).* data::= integer(0)
next_period (data)	Cette méthode est utilisée pour déclencher la fin régulière (et le redémarrage) d'une période. Cloture (met fin à) la période de mesure en cours. Met à jour capture_time et start_time et copie current_average_value dans last_average_value, met current_average_value à sa valeur par défaut. Démarre la prochaine

période de mesure.

REMARQUE L'ancienne valeur de "old last_average_value " (et de "capture_time") peut être lue pendant la durée “period”. L'ancienne valeur de "old current_average_value" n'est plus disponible à l'interface.

data::= integer(0)

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

Cette IC permet de modéliser la gestion des différentes structures de tarification. Pour chaque objet “Register activation”, des groupes d'objets “Register”, “Extended register” ou “Demand register”, modélisation de différentes sortes de grandeurs (par exemple: énergie active, puissance active, énergie réactive, etc.) sont attribués. Des sous-groupes de ces registres, définis par les *activation masks* (masques d'activation) définissent différentes structures de tarifs (par exemple: tarif de jour, tarif de nuit). Un de ces masques d'activation, le *active mask* (masque actif), définit quel sous-ensemble des registres, attribués à l'instance d'objet “Register activation”, est actif. Les registres qui ne sont pas inclus dans l'attribut *register_assignment* de tout objet “Register activation” sont toujours activés par défaut.

Register activation (activation de registre)		**0...n**	**class_id = 6, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. register_assignment	(static)	array				x + 0x08
3. mask_list	(static)	array				x + 0x10
4. active_mask	(dyn.)	octet-string				x+ 0x18
Méthodes spécifiques		***m/o***				
1. add_register (data)		o				x + 0x30
2. add_mask (data)		o				x + 0x38
3. delete_mask (data)		o				x + 0x40

Description d'attribut

logical_name Identifie l'instance de l'objet “Register activation” (Activation de registre). Voir 6.2.10.

register_assignment Spécifie une liste ordonnée d'objets COSEM attribués à l'objet “Register activation”. La liste peut contenir différentes sortes d'objets COSEM (par exemple: “Register”, “Extended register” ou “Demand register”).

array object_definition

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

mask_list Spécifie une liste de masques d'activation de registres. Chaque entrée (masque) est identifiée par son mask_name et contient un tableau d'indices renvoyant aux registres attribués au masque (le premier objet dans register_assignment est référencé par l'indice 1,

	le deuxième objet par l'indice 2,...).
	
	array register_act_mask
	
	register_act_mask::= structure
	
	{
	mask_name: octet-string,
	index_list: index_array
	}
	mask_name doit être défini de manière unique au sein de l'objet.
	index_array::= array unsigned

active_mask	Définit le masque actuellement actif. Le masque est défini par son mask_name (voir mask_list). L'active_mask définit les registres qui sont actuellement activés; tous les autres registres énumérés dans le register_assignment sont désactivés.
Description de la méthode	

add_register (data)	Ajoute un registre de plus à l'attribut register_assignment. Le nouveau registre est ajouté à la fin du tableau, à savoir le registre nouvellement ajouté possède l'indice le plus élevé. Les indices des registres existants ne sont pas modifiés.
	
	data::= structure
	{
	class_id: long-unsigned,
	logical_name: octet-string
	}

add_mask (data)	Ajoute un autre masque à l'attribut mask_list. S'il existe déjà un masque ayant le même nom, le masque existant est écrasé par le nouveau masque. data::= register_act_mask (voir ci-dessus)
delete_mask (data)	Supprime un masque de l'attribut mask_list. Le masque est défini par son nom de masque. data::= octet-string (mask_name)

5.2.6 Profil générique (class_id: 7, version: 1)

Cette IC fournit un concept généralisé permettant de stocker, de trier et d'accéder à des groupes de données ou séries de données, appelés *capture objects* (objets de saisie). Les objets de saisie sont des attributs ou éléments appropriés d'un ou de plusieurs attributs d'objets COSEM. Les objets de saisie sont recueillis de façon périodique ou occasionnelle.

Un profil est un *buffer* (tampon) pour stocker les données saisies. Pour récupérer une partie seulement du tampon, il est permis de spécifier soit une plage de valeurs, soit une plage d'entrées, demandant de récupérer toutes les entrées qui s'inscrivent dans la plage spécifiée.

La liste des *capture objects* définit la valeur à stocker dans le *buffer* (en utilisant la saisie automatique ou la méthode *capture*). La liste est définie de façon statique afin d'assurer des entrées de tampon homogènes (toutes les entrées ont la même taille et la même structure). Si la liste d'objets de saisie est modifiée, le tampon est vidé. Si le tampon est saisi par d'autres objets ""“Profile generic” (profil générique), leur tampon est vidé également, afin de garantir l'homogénéité de leurs entrées de tampon.

Le tampon peut être défini en étant trié suivant l'un des *capture objects* (objets de saisie), par exemple l'horloge ou bien les entrées peuvent être empilées (dans l'ordre "dernier entré premier sorti"). Par exemple, il est très facile de construire un "registre de puissance maximum" avec une capture de profil trié d'une profondeur d'une entrée et trié par un attribut *last_average_value* de “Demand register”. Il est tout aussi simple de définir un profil retenant les trois valeurs les plus élevées d'une certaine période.

La taille des données du profil est déterminée par trois paramètres:

a) le nombre d'entrées remplies (*entries_in_use*). Il sera de zéro après effacement du profil;

b) le nombre maximum d'entrée à conserver (*profile_entries*). Si toutes les entrées sont remplies et une demande capture () se produit, l'entrée la moins importante (selon la méthode de tri demandée) sera perdue. Ce nombre maximum d'entrées peut être spécifié. Lorsqu'il change, le tampon sera réajusté;

c) la limite physique du tampon. Cette limite dépend typiquement des objets à saisir. L'objet rejettera une tentative d'établissement du nombre maximum d'entrées à une valeur supérieure à celle physiquement possible.

Profil générique		**0...n**	**class_id = 7, version = 1**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. buffer	(dyn.)	compact-array ou array				x + 0x08
3. capture_objects	(static)	array				x + 0x10
4. capture_period	(static)	double-long-unsigned				x + 0x18
5. sort_method	(static)	enum				x + 0x20
6. sort_object	(static)	capture_object_ definition				x + 0x28
7. entries_in_use	(dyn.)	double-long-unsigned	0		0	x + 0x30
8. profile_entries	(static)	double-long-unsigned	1		1	x + 0x38
Méthodes spécifiques		***m/o***				
1. reset (data)		O				x + 0x58
2. capture (data)		O				x + 0x60
3. *reserved from previous versions* (réservé provenant de versions précédentes)		O				
4. *reserved from previous versions* (réservé provenant de versions précédentes)		O				

Description d'attribut

logical_name Identifie l'instance de l'objet “Profile generic” (Profil générique). Pour des exemples, voir 6.2.15, 6.2.28, 6.2.31, 6.2.33, 6.2.44, 6.2.46, 6.3.2, etc.

buffer Contient une séquence d'entrées. Chaque entrée contient des valeurs

des objets saisis (telles qu'elles seraient retournées par un GET.request ou un Read. request).

entrée compact-array ou array

entry::= structure

```
{
        CHOICE
        {
            -- types de données simples
            null-data                   [0],
            boolean                     [3],
            bit-string                  [4],
            double-long                 [5],
            double-long-unsigned        [6],
            octet-string                [9],
            visible-string              [10],
            UTF8-string                 [12],
            bcd                         [13],
            integer                     [15],
            long                        [16],
            unsigned                    [17],
            long-unsigned               [18],
            long64                      [20],
            long64-unsigned             [21],
            enum                        [22],
            float32                     [23],
            float64                     [24],
            date-time                   [25],
            date                        [26],
            time                        [27],
            -- types de données complexes
            array                       [1],
            structure                   [2],
            compact-array               [19]
        }
}
```

Le nombre et l'ordre des éléments de la structure contenant les entrées sont les mêmes que dans la définition des capture_objects. Le tampon est rempli par des saisies automatiques ou par des appels ultérieurs à la méthode (capture). La séquence des entrées dans la matrice est ordonnée selon la méthode de tri spécifiée.

Par défaut: Le tampon est vide après une réinitialisation.

REMARQUE 1 La lecture du tampon tout entier ne donne que les entrées qui sont "en utilisation".

REMARQUE 2 La valeur d'un objet saisi peut être remplacée par des données "null"si elle peut être récupérée sans ambiguïté à partir de la valeur précédente (par exemple pour le temps: si elle peut être calculée à partir de la valeur précédente et de capture_period; ou pour une valeur: si elle est égale à la valeur précédente).

l'**accès sélectif** (voir 4.4) à l'attribut buffer (tampon) peut être disponible (en option). Les paramètres d'accès sélectif sont tels que définis ci-dessous.

capture_objects Spécifie la liste des objets de saisie qui sont attribués à l'instance d'objet.

Sur appel de la méthode capture (data) ou automatiquement dans des intervalles définis, les attributs sélectionnés sont copiés dans le

tampon du profil.

array capture_object_definition

capture_object_definition::= structure

{

class_id:	long-unsigned,
logical_name:	octet-string,
attribute_index:	integer,
data_index:	long-unsigned

}

- où attribute_index est un pointeur vers l'attribut au sein de l'objet, attribute_index 1 se rapporte au 1er attribut (à savoir logical_name, le nom logique), attribute_index 2 au 2ème, etc.); attribute_index 0 se réfère à tous les attributs publics;
- où data_index est un pointeur sélectionnant un élément spécifique de l'attribut. Le premier élément dans l'attribut structure est identifié par data_index 1. Si l'attribut n'est pas une structure, le data_index n'a pas de signification. Si l'objet de saisie est le tampon d'un profil, le data_index identifie l'objet saisi du tampon (à savoir la colonne) du profil interne.
- data_index 0: référence l'attribut tout entier.

capture_period

>= 1: Saisie automatique supposée. Spécifie la période de saisie en secondes.

0: Absence de saisie automatique; la saisie est déclenchée extérieurement ou des événements de saisie se produisent de manière asynchrone.

sort_method

Si le profil n'est pas trié, il fonctionne comme un tampon "premier entré, premier sorti" (il est donc trié par saisie et pas nécessairement par le temps maintenu dans l'objet Clock). Si le tampon est plein, le prochain appel de capture () expulsera la première (la plus ancienne) entrée du tampon pour dégager de la place pour la nouvelle entrée.

Si le profil est trié, un appel de capture () stockera la nouvelle entrée à l'emplacement approprié dans le tampon, déplaçant toutes les entrées suivantes et perdant probablement l'entrée la moins intéressante. Si la nouvelle entrée entrait dans le tampon après la dernière entrée et si le tampon est déjà plein, la nouvelle entrée ne sera pas conservée du tout.

enum:	(1)	fifo (first in first out, c'est-à-dire première entrée, première sortie)
	(2)	lifo (last in first out, c'est-à-dire dernière entrée, première sortie)
	(3)	largest (c'est-à-dire la plus grande),
	(4)	smallest (c'est-à-dire la plus petite),
	(5)	nearest_to_zero (c'est-à-dire la plus proche de zéro),
	(6)	farest_from_zero (la plus éloignée de zéro)
Def.	fifo	

sort_object

Si le profil est trié, cet attribut spécifie le registre ou l'horloge servant de base à l'ordre de classement.

	capture_object_ definition	Voir ci-dessus.
	Def.	aucun objet par lequel trier (possible seulement possible avec sort_method fifo ou lifo)

NOTE 1 Si la sort_method est FIFO ou LIFO, tous les éléments de la capture_object_definition spécifiant l'objet de tri peuvent être zéro.

entries_in_use	Compte le nombre d'entrées stockées dans le tampon. Après un appel de la méthode reset (), le tampon ne contient pas d'entrées et cette valeur est zéro. Sur chaque appel ultérieur de la méthode capture (), cette valeur sera incrémentée jusqu'au nombre maximum d'entrées qui seront stockées (voir profile_entries).	
	double-long-unsigned	0…profile_entries
	Def.	0
profile_entries	Spécifie le nombre des entrées qui doivent être conservées dans le tampon.	
	double-long-unsigned	1…(limité par la taille physique)
	Def.	1

Paramètres pour l'accès sélectif à l'attribut tampon

Sélecteur d'accès	Paramètre d'accès	Commentaire
1	range_descriptor	Seuls les éléments de tampon correspondant au range_descriptor doivent être retournés dans la réponse.
2	entry_descriptor	Seuls les éléments de tampon correspondant à l'entry_descriptor doivent être retournés dans la réponse.

range_descriptor::= structure

{

restricting_object: capture_object_definition	Définit le capture_object limitant la plage des entrées à récupérer. Seuls les types de données simples sont autorisés.
from_value:	L'entrée à récupérer la plus ancienne ou la plus petite

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

to_value: CHOICE {voir ci-dessus} — Entrée à récupérer la plus récente ou la plus grande

selected_values: array capture_object_definition

Liste de colonnes à récupérer. Si le tableau est vide (n'a pas d'entrées), toutes les données saisies sont retournées. Autrement, seules les colonnes spécifiées dans le tableau sont retournées. Le type *capture_object_definition* est spécifié ci-dessus (*capture_objects*).

}

entry_descriptor::= structure

{

from_entry:	double-long-unsigned	première entrée à récupérer
to_entry:	double-long-unsigned	dernière entrée à récupérer to_entry == 0: l'entrée la plus élevée possible,
from_selected_value:	long-unsigned	indice de la première valeur à récupérer,
to_selected_value:	long-unsigned	indice de la dernière valeur à récupérer to_selected_value == 0: selected_value la plus élevée possible

}

NOTE 2 from_entry et to_entry identifient les lignes, from_selected_value to_selected_value identifie les colonnes du tampon à récupérer.

NOTE 3 La numérotation des entrées et des valeurs selectionnées commence à 1.

Description de la méthode	
reset (data)	Vide le tampon. Le tampon n'a pas d'entrées valides après son exécution; entries_in_use est zéro après cet appel. Cet appel ne déclenche pas d'opérations supplémentaires d'objets de saisie. Précisément, il ne réinitialise pas de tampons ou registres saisis.

	data::= integer(0)
capture (data)	Copie les valeurs des objets à saisir dans le tampon en lisant chaque objet de saisie. En fonction de la sort_method et de l'état réel du tampon, elle produit une nouvelle entrée ou un remplacement pour l'entrée la moins significative. Tant que toutes les entrées n'ont pas été déjà utilisées, l'attribut entries_in_use sera incrémenté. Cet appel ne déclenche pas d'opérations supplémentaires au sein des objets de saisie telles que capture () ou reset (). A noter que, s'il est nécessaire de saisir plus d'un attribut d'un objet, ces attributs doivent être définis un à un sur la liste des objets de saisie. Si l'attribute_index = 0, tous les attributs sont saisis. data::= integer(0)

Comportement de l'objet après modification de certains attributs

Toute modification de l'un des capture_objects décrivant la structure statique du tampon appellera automatiquement un reset () et cet appel se propagera à tous les autres profils capturant ce profil.

En cas de tentative d'écriture dans profile_entries d'une valeur trop grande pour le tampon, celle-ci sera rejetée.

Restrictions

Lors de la définition des objets de saisie, toute référence circulaire au profil doit être évitée.

Profil utilisé pour définir un sous-ensemble de valeurs de lecture préférentielle

En mettant profile_entries à 1, un objet “Profile generic” peut être utilisé pour définir un ensemble de valeurs de lecture préférentielle. Voir aussi 6.2.15. La mise de capture_period à 1 assure que les valeurs sont mises à jour toutes les secondes.

5.2.7 Table de fournisseur de service d’énergie (class_id: 26, version: 0)

Cette IC permet d'encapsuler des données en tables de l’ANSI C12.19. Chaque "table" est représentée par une instance de cette IC, identifiée par son *nom logique*.

Tableaux d'utilités	**0...n**	**class_id = 26, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. table_ID (static)	long-unsigned				x + 0x08
3. length	double-long-unsigned				x + 0x10
4. buffer	octet-string				x + 0x18
Méthodes spécifiques	***m/o***				

Description d'attribut

logical_name	Identifie l'instance de l'objet “Utility tables” (Table d’utility). Voir 6.2.27
table_ID	Numéro de table. Numéro de tableau est tel que spécifié dans la

	norme ANSI et peut être soit un tableau standard, soit un tableau de fabricant.
d) length	Nombre d'octets dans le tampon du tableau.
e) buffer	Contenu de table. **L'accès sélectif** (voir 4.4) à l'attribut *buffer* peut être disponible (en option). Les paramètres d'accès sélectif sont tels que définis ci-dessous.

Paramètres pour l'accès sélectif à l'attribut tampon

Sélecteur d'accès	Paramètre	Commentaire
1	offset_access	Accès à la table par offset (c'est-à-dire décalage) et count (c'est-à-dire compteur) utilisant l'offset_selector pour les données de paramètre.
2	index_access	Accès à la table par id d'élément et nombre d'éléments en utilisant index_selector pour les données de paramètre.

offset_selector::= structure

{

Offset:	double-long-unsigned	décalage en octets au début de la zone d'accès, par rapport au début du tableau
Count:	long-unsigned	nombre d'octets demandés ou transférés

}

index_selector::= structure

{

Index:	array of long-unsigned	séquence d'indices pour identifier des éléments dans la hiérarchie du tableau
Count:	long-unsigned	nombre d'éléments demandés ou transférés Les valeurs de "count" supérieures à 1 retournent autant d'éléments au maximum. Une valeur de zéro, lorsqu'elle est donnée dans le contexte d'une demande, se réfère au sous-arbre entier de la hiérarchie commençant au point de sélection.

}

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

Cette IC permet de regrouper des entrées homogènes, des attributs identiques d'objets multiples, qui sont tous des instances de la même IC. De plus, dans leur nom logique (code OBIS), la valeur dans les groupes de valeurs A à D et F est identique. Les valeurs possibles dans le groupe de valeurs E sont définies dans la CEI 62056-6-1 sous une forme tabulée: l'en-tête de tableau définit la partie commune du code OBIS et chaque cellule du tableau définit une valeur possible du groupe de valeurs E. Un objet “Register table” (Table de registre) peut saisir les attributs d'un certain nombre ou de la totalité de ces objets.

NOTE 1 Certains exemples sont le tableau “UNIPEDE voltage dip quantities” (Grandeurs de creux de tension UNIPEDE), voir CEI 62056-6-1:—, Tableau 18, ou le tableau “Extended phase angle measurement” (Mesure d'angle de phase étendu), voir CEI 62056-6-1:—, Tableau 16.

NOTE 2 Si une fonctionnalité plus complexe est nécessaire, l'IC “Profile generic” peut être utilisée.

Tableau de registre		0...n	class_id = 61, version = 0			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. table_cell_values	(dyn.)	array				x + 0x08
3. table_cell_definition	(static)	structure				x + 0x10
4. scaler_unit	(static)	scaler_unit_type				x + 0x18
Méthodes spécifiques		***m/o***				
1. reset (data)		o				x + 0x28
2. capture (data)		o				x + 0x30

Description d'attribut

logical_name

Identifie l'instance de l'objet “Register table ” (Tableau de registre).

Lorsque le format du nom logique est A.B.C.D.255.F; les valeurs A à D et F définissent la partie commune du nom logique des objets, dont les attributs sont saisis. Un seul attribut des objets concernés peut être saisi (par exemple l'attribut *value*).

Lorsque le format du nom logique est A.B.98.10.x.255, plusieurs instances de l'IC “Register table” peuvent être utilisées pour saisir différents attributs des objets concernés. Le groupe de valeurs E numérote les instances. Voir CEI 62056-6-1:—, 6.4.

table_cell_values

Contient la valeur des attributs saisis, telle qu'elle serait retournée sur une GET ou Read.request aux attributs individuels.

compact array ou array table_cell_entry

table_cell_entry

CHOICE
{
-- types de données simples
null-data [0],
bit-string [4],
double-long [5],
double-long-unsigned [6],
octet-string [9],
visible-string [10],
UTF8-string [12],
bcd [13],
integer [15],
long [16],
unsigned [17],
long-unsigned [18],
long64 [20],
long64-unsigned [21],
float32 [23],
float64 [24],
-- types de données complexes
structure [2]
}

	Si l'attribut saisi est attribute_0, les valeurs redondantes peuvent être remplacées par “null-data”, si leur valeur peut être récupérée sans ambiguïté (par exemple: scaler_unit).

table_cell_definition	Spécifie la liste des attributs saisis dans le tableau de registre.
	
	structure
	
	{
	class_id: long-unsigned,
	logical_name: octet-string,
	group_E_values: array cell_identifier,
	{
	cell_identifier: unsigned
	}
	attribute_index: integer
	}
	où:
	
	• class_id définit le class_id commun à tous les objets dont les attributs sont saisis;
	• logical_name contient le nom logique commun à tous les objets, avec E = 255 (joker);
	• les group_E_values contiennent la liste des identificateurs de cellules, tels que définis dans le tableau correspondant de la CEI 62056-6-1;
	• attribute_index est un pointeur sur l'attribut au sein de l'objet. attribute_index 0 se réfère à tous les attributs publics.
	
	Si le nom logique de l'objet “Register table” est au format A.B.C.D.255.F et les attributs définis de tous les objets identifiés dans le tableau correspondant de la CEI 62056-6-1 sont saisis, l'attribut 3 peut ne pas être accessible. Dans ce cas:
	
	• le class_id doit être 1 “Data”, 3 “Register” ou 4 “Extended register”;
	• le nom logique des objets à saisir est défini par le nom logique de l'objet “Register table” et le tableau correspondant de la CEI 62056-6-1.
	• l'indice d'attribut doit être 2 (value).

scaler_unit	Voir la description de l'IC “Register”. Lorsque les attributs “value” des objets “Register” ou “Extended register” sont saisis, l'attribut scaler_unit doit être commun à tous les objets et cet attribut doit contenir une copie. Si d'autres attributs ou IC sont saisi(e)s, l'attribut scaler_unit n'a pas de signification et doit être inaccessible.
Description de la méthode	
reset (data)	Efface les table_cell_values. Il n'a pas d'effet sur les attributs saisis.

	data::= integer(0)
capture (data)	Copie les valeurs des attributs dans les table_cell_values. Si l'attribute_index = 0, tous les attributs sont saisis.

Comportement de l'objet après modification de l'attribut table_cell_definition

Toute modification apporté à cet attribut appellera automatiquement la méthode reset(data) et elle sera propagée à tous les profils capturant cet objet.

En cas de tentative d'écriture dans table_cell_definition d'une valeur trop grande pour le tampon contenant l'attribut table_cell_values, elle sera rejetée.

5.2.9 Cartographie de statut (class_id: 63, version: 0)

Cette IC permet de modéliser la cartographie des bits dans un mot d'état à des entrées dans un tableau de référence.

Status mapping (Mise en correspondance de statut)		**0...n**	**class_id = 63, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. status_word	(dyn.)	CHOICE				x + 0x08
3. mapping_table	(static)	structure				x + 0x10
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name	Identifie les instances de l'objet “Status mapping”. Voir 6.2.31, 6.2.33, 6.2.39 et 6.3.7

status_word	Contient la valeur courante du mot d'état.
	CHOICE
	{
	bit-string [4],
	double-long-unsigned [6],
	octet-string [9],
	visible-string, [10],
	UTF8-string [12],
	unsigned [17],
	long-unsigned [18],
	long64-unsigned [21]
	}
	La taille du status_word et de n*8 bits, la taille maximale est de 65 536 bits.
	NOTE 1 Les fabricants peuvent choisir n'importe lesquels des types énumérés ci-dessus. Cependant, le mot de statut est toujours interprété comme un bit-string (chaîne de bits).

mapping_table	Contient la mise en correspondance du mot d'état aux positions dans le tableau de référence.
	
	structure
	
	{ ref_table_id: unsigned,
	CHOICE
	{
	first_entry: long-unsigned,
	array: table_entries
	}
	}
	table_entry: long-unsigned
	
	où:
	
	• ref_table_id est l'identificateur du tableau de statuts de référence.
	
	NOTE 2 Les tableaux de statuts de référence sont maintenus par la DLMS UA. Ils peuvent être spécifiques à chaque projet.
	
	• first_entry est l'entrée dans le tableau de référence qui correspond au bit 0 du mot de statut: le bit 1 correspond à l'entrée suivante et ainsi de suite, la dernière entrée étant définie par la longueur du mot de statut;
	
	• Si le choix "array" est pris, le tableau d'entrées de table met les bits du mot d'état en correspondance avec les entrées contenues dans la table de mot d'état de référence. La position dans le tableau définit la position dans le mot de statut (la première position correspond au bit 0).

5.3 Classes d'interfaces pour contrôle d'accès et gestion

5.3.1 SN d'association (class_id: 12, version: 2)

Les dispositifs logiques COSEM capables d'établir des AA au sein d'un contexte COSEM en utilisant le référencement par SN modélisent les AA à travers les instances de l'IC "SN d'association". Un dispositif logique COSEM peut avoir une instance de cette IC pour chaque AA que le dispositif est capable de prendre en charge.

Le **short_name** de l'objet "Association SN" lui-même est fixé dans le contexte COSEM. Voir 4.3.

SN d'association	0...n	class_id = 12, version = 2			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. object_list (static)	objlist_type				x + 0x08
3. access_rights_list (static)	access_rights_type				x + 0x10
4. security_setup_reference (static)	octet-string				x + 0x18
Méthodes spécifiques	***m/o***				
1. *reserved from previous versions* (réservé provenant de versions précédentes)	o				
2. *reserved from previous versions* (réservé provenant de versions précédentes)	o				
3. read_by_logicalname (data)	o				x + 0x30
4. *reserved from previous versions* (réservé provenant de versions précédentes)	o				
5. change_secret (data)	o				x + 0x40
6. *reserved from previous versions* (réservé provenant de versions précédentes)	o				
7. *reserved from previous versions* (réservé provenant de versions précédentes)					
8. reply_to_HLS_authentication (data)	o				x + 0x58

Description d'attribut

logical_name

Identifie l'instance de l'objet "Association SN" (SN d'association). Voir 6.2.22.

object_list

Contient la liste de tous les objets avec leurs base_name (short_name), class_id, version et logical_name. Le base_name est l'objectName DLMS du premier attribut (logical_name).

object_list_type::= array objlist_element

objlist_element::= structure

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

l'accès sélectif (voir 4.4) à l'attribut object_list peut être disponible (en option). Les valeurs de sélecteur d'accès et leurs paramètres sont tels que définis ci-dessous.

access_rights_list

Contient les droits d'accès aux attributs et méthodes.

La liaison entre l'object_list et l'access_rights_list est le base_name, présent à la fois dans la structure object_list_element et la structure access_right_element. Par conséquent, les base_names sur les deux listes doivent être les mêmes.

Le nombre – et de préférence, l'ordre – des éléments doit être le même dans la matrice d'object_list_element et dans la matrice

d'access_right_element.

```
access_rights_type::= array          access_rights_element

access_rights_element::= structure

{
        base_name:              long,
        attribute_access: attribute_access_descriptor,
        method_access:  method_access_descriptor
}
attribute_access_descriptor::= array                attribute_access_item

attribute_access_item::= structure
{
        attribute_id:           integer,
        access_mode:            enum,
                                {
                                    (0)    no_access,
                                    (1)    read_only,
                                    (2)    write_only,
                                    (3)    read_and_write,
                                    (4)    authenticated_read_only,
                                    (5)    authenticated_write_only,
                                    (6)    authenticated_read_and_write
                                }
        access_selectors:       CHOICE
                                {
                                    null-data,
                                    array of integer
                                }
}
method_access_descriptor::= array       method_access_item

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

                                {
                                        (0)     no_access,
                                        (1)     access,
                                        (2)     authenticated_access
                                }
}
```

l'**accès sélectif** (voir 4.4) à l'attribut access_rights_list peut être disponible (en option). Les valeurs de sélecteur d'accès et leurs paramètres sont tels que définis ci-dessous.

security_setup_ reference	Référence l'objet "Security setup" par son nom logique. L'objet référencé gère la sécurité pour une instance d'objet donnée de l'Association SN.

Paramètres pour accès sélectif pour l'attribut object_list et access_right_list

Valeur de sélecteur d'accès	Paramètre	Disponible avec l'attribut	Commentaire
1	class_id: long-unsigned	2	Délivre le sous-ensemble de l'object_list pour une class_id spécifique. Pour la réponse: data::= object_list_type

2	structure
	{
	class_id: long-unsigned,
	logical_name: octet-string
	}
		2	Délivre l'entrée de l'object_list pour un class_id et un logical_name spécifiques.
			Pour la réponse: data::= object_list_element

3	base_name: long	2, 3	Dans le cas de l'attribut 2, délivre l'entrée de l'object_list pour un base_name spécifique. Pour la réponse: data::= object_list_element Dans le cas de l'attribut 3, délivre l'entrée de l'access_rights_list pour un base_name spécifique. Pour la réponse: data::= access_rights_element

Description de la méthode

read_by_ logicalname (data)

Lit les attributs pour les objets sélectionnés. Les objets sont spécifiés par leur class_id et leur logical_name. Avec cette méthode, la caractéristique d'accès paramétré peut aussi être utilisée.

data::= array attribute_identification

attribute_identification::= structure

{

class_id: long-unsigned,
logical_name: octet-string,
attribute_index: integer

}

où attribute_index est un pointeur (à savoir, le décalage) vers l'attribut au sein de l'objet.

attribute_index 0 délivre tous les attributs [a], attribute_index 1 délivre le premier attribut (à savoir: logical_name, etc.).

Pour la réponse: les données sont selon le type de l'attribut.

change_secret (data)

Change le secret LLS ou HLS (par exemple: mot de passe).

data::= octet-string new secret[b] (c'est-à-dire nouveau secret)

NOTE Dans le cas de HLS avec GMAC, le (HLS_)secret est détenu par l'objet "Security setup" référencé dans l'attribut 4.

reply_to_HLS_ authentication (data)

L'invocation à distance de cette méthode délivre au serveur le résultat du traitement de secret par le client du défi du serveur au client, f(StoC), comme le paramètre de service data (donnée) de la primitive Read.request invoquée avec l'accès paramétré.

data::= octet-string réponse du client au défi

Si l'authentification est acceptée, la réponse (primitive Read.confirm) contient Result == OK et le résultat du traitement de secret par le serveur du défi du client au serveur, f(CtoS) dans le paramètre de

service data (données) du service Read.response.

data::= octet-string réponse du service au défi

Si l'authentification n'est pas acceptée, le paramètre résultat dans la réponse doit contenir une valeur non-OK et aucune donnée ne doit être renvoyée.

[a] Si au moins un attribut n'a pas de droit d'accès en lecture dans le cadre de l'association courante, un read_by_logicalname() à l'indice d'attribut 0 révèle un message d'erreur “champ d'accès violé”, voir CEI 62056-5-3:—, Article 8.

[b] La structure du “new secret” dépend du mécanisme de sécurité implémenté. Le “new secret” peut contenir des bits de contrôle supplémentaires et il peut être chiffré.

5.3.2 Association LN (class_id: 15, version: 1)

Les dispositifs logiques COSEM capables d'établir des AA au sein d'un contexte COSEM en utilisant le référencement par LN modélisent les AA à travers les instances de l'IC “LN d'association”. Un dispositif logique COSEM a une instance de cette IC pour chaque AA que le dispositif est capable de prendre en charge.

LN d'association		**0...MaxNbofAss.**	**class_id = 15, version = 1**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. object_list	(static)	object_list_type				x + 0x08
3. associated_partners_id		associated_partners_type				x + 0x10
4. application_context_name		application_context_name				x + 0x18
5. xDLMS_context_info		xDLMS_context_type				x + 0x20
6. authentication_mechanism_name		mechanism_name				x + 0x28
7. secret		octet-string				x + 0x30
8. association_status		enum				x + 0x38
9. security_setup_reference	(static)	octet-string				x + 0x40
Méthodes spécifiques		***m/o***				
1. reply_to_HLS_authentication (data)		o				x + 0x60
2. change_HLS_secret (data)		o				x + 0x68
3. add_object (data)		o				x + 0x70
4. remove_object (data)		o				x + 0x78

Description d'attribut

logical_name Identifie l'instance de l'objet “Association LN” (LN d'association). Voir 6.2.22.

object_list Contient la liste des objets COSEM visibles avec leurs class_id, version, nom logique et les droits d'accès à leurs attributs et méthodes au sein de l'association d'application donnée.

object_list_type::= array object_list_element

object_list_element::= structure

{

class_id: long-unsigned,

```
        version:                unsigned,
        logical_name:           octet-string,
        access_rights:          access_right
}
access_right::= structure
{
        attribute_access:       attribute_access_descriptor,
        method_access:          method_access_descriptor
}
attribute_access_descriptor::= array    attribute_access_item

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

                                {
                                    (0)   no_access,
                                    (1)   read_only,
                                    (2)   write_only,
                                    (3)   read_and_write,
                                    (4)   authenticated_read_only,
                                    (5)   authenticated_write_only,
                                    (6)   authenticated_read_and_write
                                }
        access_selectors:       CHOICE
                                {
                                    null-data,
                                    array of integer
                                }
}
method_access_descriptor::= arraymethod_access_item

method_access_item::= structure
{
        method_id: integer,
        access_mode:    enum
                        {
                            (0)   no_access,
                            (1)   access,
                            (2)   authenticated_access
                        }
}
```

où:

- l'attribute_access_descriptor et le method_access_descriptor contiennent toujours tous les attributs ou toutes les méthodes implémenté(e)s;
- les access_selectors contiennent une liste des valeurs de sélecteur prises en charge.

l'accès sélectif (voir 4.4) à l'attribut object_list peut être disponible (en option). Les paramètres d'accès sélectif sont tels que définis ci-dessous.

associated_partners_id	Contient les identificateurs des AP (de dispositif logique) client et serveur COSEM au sein des dispositifs physiques hébergeant ces AP, qui appartiennent à l'AA modélisée par l'objet “Association LN” (LN d'association). associated_partners_type::= structure {

client_SAP: integer,
server_SAP: long-unsigned
}

La plage pour le client_SAP est 0…0x7F.

La plage pour le server_SAP est 0x0000…0x3FFF.

Les SAP doivent se situer dans la plage permise par le type de données et les supports.

application_context_name

Dans l'environnement COSEM, il est prévu qu'un contexte d'application préexiste et soit référencé par son nom pendant l'établissement d'une AA. Cet attribut contient le nom du contexte d'application pour l'association en question.

CHOICE
{
structure [2],
octet-string [9]
}

Le nom du contexte d'application est spécifié comme appartenant au type OBJECT IDENTIFIER dans la CEI 62056-5-3:—, 7.2.2.2. L'attribut application_context_name inclut les "arc labels" (étiquettes d'arc) de l'OBJECT IDENTIFIER. Dans ce cas:

application_context_name::= structure
{
joint-iso-ctt-element: unsigned,
country-element: unsigned,
country-name-element: long-unsigned,
identified-organization-element: unsigned,
DLMS-UA-element: unsigned,
application-context-element: unsigned,
context-id-element: unsigned
}

Pour les mises en œuvre existantes, l'attribut peut contenir la valeur de l'OBJECT IDENTIFIER codée en BER, comme une chaîne d'octets. Voir CEI 62056-5-3:—, B.4. Dans ce cas:

application_context_name::= octet-string

// contient la valeur de l'identificateur d'OBJECT codé BER

Exemples

Dans le cas du context_id(1), le codage A-XDR en tant que structure (toutes les valeurs sont hexadécimales): 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01.

Le codage A-XDR en chaîne d'octets, contenant la valeur de l'OBJECT IDENTIFIER codé en BER (toutes les valeurs sont hexadécimales): 09 07 60 85 74 05 08 01 01.

xDLMS_context_info

Contient toutes les informations nécessaires sur le contexte xDLMS pour l'association donnée.

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
}

où:

- l'élément conformance (conformité) contient le bloc de conformité xDLMS pris en charge par le serveur;
- l'élément max_receive_pdu_size contient la longueur maximale pour une APDU xDLMS, exprimée en octets que le client peut envoyer. Il est le même que le paramètre server-max-receive-pdu-size de l'APDU InitiateResponse xDLMS;
- l'élément max_send_pdu_size, dans une AA active, contient la longueur maximale pour une APDU xDLMS, exprimée en octets que le serveur peut envoyer. Il est le même que le paramètre client-max-receive-pdu-size de l''APDU InitiateResponse xDLMS;
- l'élément dlms_version_number contient le numéro de version de DLMS pris en charge par le serveur;
- l'élément quality_of_service n'est pas utilisé;
- le cyphering_info, dans une association active, contient le paramètre clé de l'APDU InitiateRequest xDLMS. Voir CEI 62056-5-3:—, Article 8.

authentication_ mechanism_name

Contient le nom du mécanisme d'authentification pour l'association.

```
mechanism-name      CHOICE
{
        structure    [2],
        octet-string [9]
}
```

Le nom du mécanisme d'authentification est spécifié comme appartenant au type OBJECT IDENTIFIER dans la CEI 62056-5-3:—, 7.2.2.3.

L'attribut authentication_mechanism_name inclut les "arc labels" (étiquettes d'arc) de l'OBJECT IDENTIFIER. Dans ce cas:

```
authentication_mechanism_name::= structure
{
        joint-iso-ctt-element:                  unsigned,
        country-element:                        unsigned,
        country-name-element:                   long-unsigned,
        identified-organization-element:        unsigned,
        DLMS-UA-element:                        unsigned,
        authentication-mechanism-name-element:  unsigned,
        mechanism-id-element:                   unsigned
}
```

Pour les mises en œuvre existantes, l'attribut peut contenir la valeur de l'OBJECT IDENTIFIER codée en BER, comme une chaîne d'octets. Voir CEI 62056-5-3:—, B.4. Dans ce cas:

authentication_mechanism_name::= octet-string

// contient la valeur de l'identificateur d'OBJECT codé BER

Exemples:

Dans le cas du mechanism_id(1), le codage A-XDR en tant que structure (toutes les valeurs sont hexadécimales): 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01.

Le codage A-XDR en chaîne d'octets, contenant la valeur de l'OBJECT IDENTIFIER codé en BER (toutes les valeurs sont hexadécimales): 09 07 60 85 74 05 08 02 01.

	Aucun mechanism-name n'est requis lorsqu'il n'est pas utilisé d'authentification.
secret	Contient le secret pour le processus d'authentification LLS ou HLS. NOTE Dans le cas de HLS avec GMAC, le (HLS_)secret est détenu par l'objet "Security setup" référencé dans l'attribut 9.
association_status	Indique le statut courant de l'association, qui est modélisé par l'objet.
	enum: (0) non-associated, (1) association-pending, (2) associated
security_setup_ reference	Référence l'objet "Security setup" par son nom logique. L'objet référencé gère la sécurité pour une instance d'objet donnée de l'Association LN.

Une opération SET sur un attribut d'un objet "association LN" devient effective lorsque cet objet association est utilisé pour établir une nouvelle association.

Paramètres pour l'accès sélectif à l'attribut object_list

- Si aucun accès sélectif n'est demandé, (aucun paramètre Access_Selection_Parameters n'est présent dans la primitive de service GET.request (.indication) pour l'attribut object_list), le service.response (.confirmation) correspondant doit contenir tous les object_list_elements de l'attribut object_list.
- Lorsqu'un accès sélectif est demandé à l'attribut object_list (le paramètre Access_Selection_Parameter est présent), la réponse doit contenir une liste "filtrée" d'object_list_elements, comme suit:

Sélecteur d'accès	Paramètre d'accès	Commentaire
1	NULL	Toutes les informations, à l'exclusion des access_rights, doivent être incluses dans la réponse.
2	class_list	Accès par class_id. Dans ce cas, seuls doivent être inclus dans la réponse les object_list_elements de l'object_list qui ont un class_id égal à l'un des class_id de la class_list. Aucune information access_right n'est incluse. class_list::= array class_id class_id: long-unsigned

3	object_id_list	Accès par objet. L'enregistrement d'informations complet des instances d'objet sur l'object_Id_list doit être retourné.
		object_id_list::= array object_id
		object_id::= structure
		{
		class_id: long-unsigned,
		logical_name: octet-string
		}

4	object_id	L'enregistrement d'informations complet de l'instance d'objet COSEM requise doit être retourné. object_id::= structure Voir ci-dessus.

Description de la méthode

reply_to_HLS_ authentication	L'invocation à distance de cette méthode délivre au serveur le résultat du traitement de secret par le client du défi du serveur au

(data)	client, f(StoC), comme le paramètre de service *data* (données) de la primitive ACTION.request invoquée. data::= octet-string réponse du client au défi Si l'authentification est acceptée, la réponse (primitive ACTION.confirm) contient Result == OK et le résultat du traitement de secret par le serveur du défi du client au serveur, f(CtoS) dans le paramètre de service *data* (données) du service de réponse. data::= octet-string réponse du service au défi Si l'authentification n'est pas acceptée, le paramètre résultat dans la réponse doit contenir une valeur non-OK et aucune donnée ne doit être renvoyée.
change_HLS_secret (data)	Change le secret HLS (par exemple: clé de chiffrement). data::= octet-string[a] nouveau secret HLS
add_object (data)	Ajoute l'objet référencé à l'object_list. data::= object_list_element (voir ci-dessus)
remove_object (data)	Retire l'objet référencé de l'object_list. data::= object_list_element (voir ci-dessus)

[a] La structure du "new secret" dépend du mécanisme de sécurité implémenté. Le "new secret" peut contenir des bits de contrôle supplémentaires et il peut être chiffré.

5.3.3 SAP assignment (class_id: 17, version: 0)

Cette IC permet de modéliser la structure logique des dispositifs physiques, en fournissant des informations relatives à l'affectation des dispositifs logiques à leurs SAP. Voir CEI 62056-5-3:—, Annexe A.

SAP assignment (Affectation de SAP)	**0...1**	**class_id = 17, version = 0**			
Attribut(s)	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. SAP_assignment_list (static)	asslist_type			0	x + 0x08
Méthodes spécifiques	***m/o***				
1. connect_logical_device (data)	o				x + 0x20

Description d'attribut

logical_name	Identifie l'instance des objets "SAP assignment" (affectation de SAP). Voir 6.2.23.

SAP_assignment_ list	Contient la liste de tous les dispositifs logiques et leurs adresses de SAP au sein du dispositif physique.
	asslist_type::= array asslist_element
	asslist_element::= structure
	{
	SAP: long-unsigned,
	logical_device_name: octet-string CHOICE
	{
	octet-string [9],
	visible-string [10],
	Unicode-UTF8 [12]
	}
	}
	REMARQUE: L'adressage effectif est accompli par les couches communication en place.

Description de la méthode

connect_logical_ device (data)	Connecte un dispositif logique à un SAP. La connexion au SAP 0 déconnectera le dispositif. Un seul dispositif peut être connecté à un SAP donné (à l'exception du SAP 0). data::= asslist_element

5.3.4 Image transfer (class_id: 18, version: 0)

5.3.4.1 Généralités

Les instances de l'IC "Image transfer" (transfert d'Image) modélisent le mécanisme de transfert de fichiers binaires, appelés Images de firmware, vers les serveurs COSEM. Le transfert d'Image s'accomplit en plusieurs étapes:

Etape 1: Le client récupère individuellement l'ImageBlockSize de chaque serveur;

Étape 2: Le client déclenche le processus de transfert d'Image, de manière individuelle ou en utilisant la diffusion;

Étape 3: Le client transfère les ImageBlock vers un (groupe de) serveur(s), de manière individuelle ou en utilisant la diffusion;

Étape 4: Le client vérifie la complétude de l'Image dans chaque serveur de manière individuelle et transfère tous les ImageBlock non (encore) transférés;

Étape 5: L'Image est vérifiée;

Étape 6: Avant activation, l'Image est validée

Étape 7: L'/les Image(s) est/sont activée(s).

5.3.4.2 Définitions relatives au processus de transfert d'Image

5.3.4.2.1
Image
donnée binaire de taille spécifiée qui peut être transférée, vérifiée et activée

Note 1 à l'article: Les données d'Image transférées et l'/les Image(s) qui sera/seront activée(s) ne sont pas nécessairement identiques.

5.3.4.2.2
ImageSize
taille de l'Image en octets

Note 1 à l'article: L'Image peut être transférée en des ImageBlock.

5.3.4.2.3
ImageBlock
partie de l'Image de la taille ImageBlockSize

Note 1 à l'article: Chaque bloc est identifié par son ImageBlockNumber.

5.3.4.2.4
ImageBlockSize
taille de l'ImageBlock en octets

5.3.4.2.5
ImageBlockNumber
identificateur d'un ImageBlock

Note 1 à l'article: Les ImageBlocks sont numérotés séquentiellement, à partir de 0.

La signification des définitions ci-dessus est illustrée dans la Figure 9.

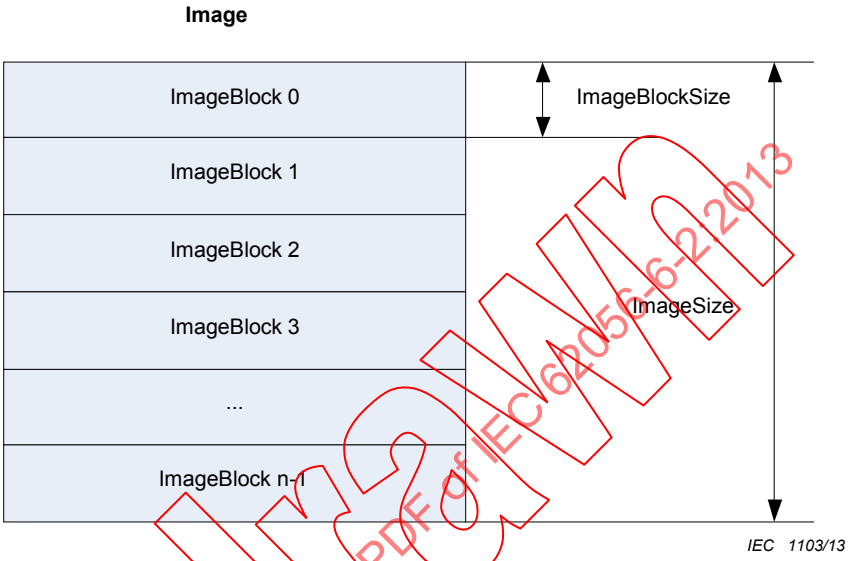


Figure 9 – Signification des définitions concernant l'Image

5.3.4.3 Le mécanisme de transfert d'Image

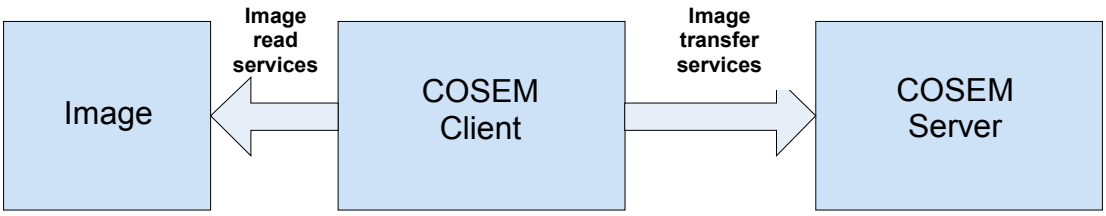
Le mécanisme de transfert d'Image consiste en deux jeux de services clients COSEM; voir Figure 10:

- Les services "Image read" (lecture d'Image) sont utilisés par le client COSEM pour lire des ImageBlock de l'Image;

NOTE 1 La création de l'Image, son emplacement et les services de lecture d'Image ne s'inscrivent pas dans le domaine d'application de la présente norme.

- Les services "Image transfer" (transfert d'Image) sont utilisés par le client COSEM pour transférer l'image vers le serveur COSEM.

Les services de transfert d'image sont corrélés avec les services COSEM, accédant aux attributs et méthodes d'un ou de plusieurs objets d'interface de transfert d'Image COSEM.



Légende

Anglais	Français
Image read services	Services lecture d'image
Image transfer services	Services transfert d'image
COSEM Client	Client COSEM
COSEM Server	COSEM Serveur

Figure 10 – Les services Image Read et Image Transfer

Le processus de transfert d'Image ne peut être lancé que s'il est activé dans le serveur.

Le processus, montré à la Figure 11, est expliqué ci-dessous.

Étape 1: Récupérer l'ImageBlockSize: La première étape consiste à récupérer l'ImageBlockSize pris en charge par le serveur, afin que le client puisse transférer des blocs d'image ayant la taille correcte. Le paramètre ImageBlockSize est détenu par l'attribut *image_block_size* et peut être différent dans des serveurs différents.

NOTE 2 Normalement, la valeur de cet attribut ne peut pas être écrite par le client: elle est une propriété du serveur.

Si les ImageBlock sont destinés à être envoyés, par diffusion, vers un groupe de serveurs COSEM, l'ImageBlockSize doit être le même dans chaque membre du groupe.

Étape 2: Déclencher le transfert d'Image: Dans la deuxième étape, le client COSEM déclenche le transfert d'Image. Le déclenchement est accompli en invoquant la méthode *image_transfer_initiate*. Le paramètre d'invocation de la méthode contient l'identificateur et la taille de l'Image devant être transférée. Le serveur doit rendre disponible l'espace mémoire nécessaire pour contenir l'Image. Le déclenchement du transfert d'Image peut être accompli individuellement ou par diffusion.

Après un déclenchement réussi, la valeur de l'attribut *image_transfer_status* est (1) *Image transfer initiated* (transfert d'Image déclenché) et le serveur COSEM est prêt pour recevoir les ImageBlock.

La valeur de l'attribut *image_to_activate_info* doit être réinitialisée.

Étape 3: Transférer les ImageBlock: Dans la troisième étape, le client – après avoir lu les ImageBlock de la taille ImageBlockSize dans l'Image – transfère ces ImageBlock vers le(s) serveur(s). Les ImageBlock peuvent être transférés individuellement ou être diffusés.

Le transfert est accompli en invoquant la méthode *image_block_transfer*. Les paramètres d'invocation de la méthode comprennent l'ImageBlockNumber et un ImageBlock. Les ImageBlock sont acceptés par les seuls serveurs COSEM dans lesquels le processus de transfert d'Image a été déclenché avec succès. Les autres serveurs rejettent silencieusement tous les éventuels ImageBlock reçus.

Étape 4: Vérifier la complétude de l'Image: Dans la quatrième étape, le client vérifie – prennant chaque serveur individuellement – la complétude de l'Image transférée. Si l'Image n'est pas complète, il transfère les ImageBlock non (encore) transférés. Il s'agit d'un processus itératif qui se poursuit jusqu'au transfert réussi de l'Image entière.

Pour identifier et transférer les ImageBlock non transférés, deux mécanismes sont disponibles:

- Le client peut récupérer le statut de chaque ImageBlock: soit *not transferred* (c'est-à-dire non transféré), soit *transferred* (c'est-à-dire transféré). Cela se réalise en récupérant la valeur de l'attribut *image_transferred_blocks_status*. Le client transfère ensuite les ImageBlock non (encore) transférés;
- En variante, le client peut récupérer l'ImageBlockNumber du premier bloc non transféré. Cela se réalise en récupérant la valeur de l'attribut *image_first_not_transferred_block_number.* Le client transfère ensuite cet ImageBlock non (encore) transféré;

Après cela, le client vérifie de nouveau la complétude de l'Image.

NOTE 3 Les deux stratégies peuvent être combinées librement.

Étape 5: Vérifier l'Image: Dans la cinquième étape, l'Image est vérifiée. Cela est réalisé en invoquant la méthode *image_verify* par le client. Le résultat peut être:

- success (succès), si la vérification a pu être achevée;
- temporary-failure (échec temporaire), si la vérification n'a pas pu être achevée;
- other-reason (autre raison), si la vérification a échoué.

NOTE 4 Pour les services xDLMS, les codes Action-Result / Data-Access-Result sont spécifiés dans la CEI 62056-5-3:—, Article 8.

La vérification peut aussi être déclenchée par le serveur.

Le résultat de la vérification de l'Image peut être vérifié par le client en récupérant la valeur de l'attribut *image_transfer_status*.

NOTE 5 Les conditions de vérification de l'Image ne relèvent pas du domaine d'application de la présente norme.

Étape 6: Vérifier l'Image avant activation: Dans la sixième étape, le client peut vérifier les informations sur l'Image transférée avant de l'activer.

Ces informations sont détenues par l'attribut *image_to_activate_info* de l'objet de transfert d'Image et sont générées comme un résultat de la vérification d'Image. Pour chaque Image devant être activée, cet attribut détient les paramètres: {image_size, image_identification, image_signature}.

Si ces informations ne sont pas ce qui est attendu, le client peut recommencer le transfert de l'image.

Autrement, il passe à l'étape suivante, activation de l'Image.

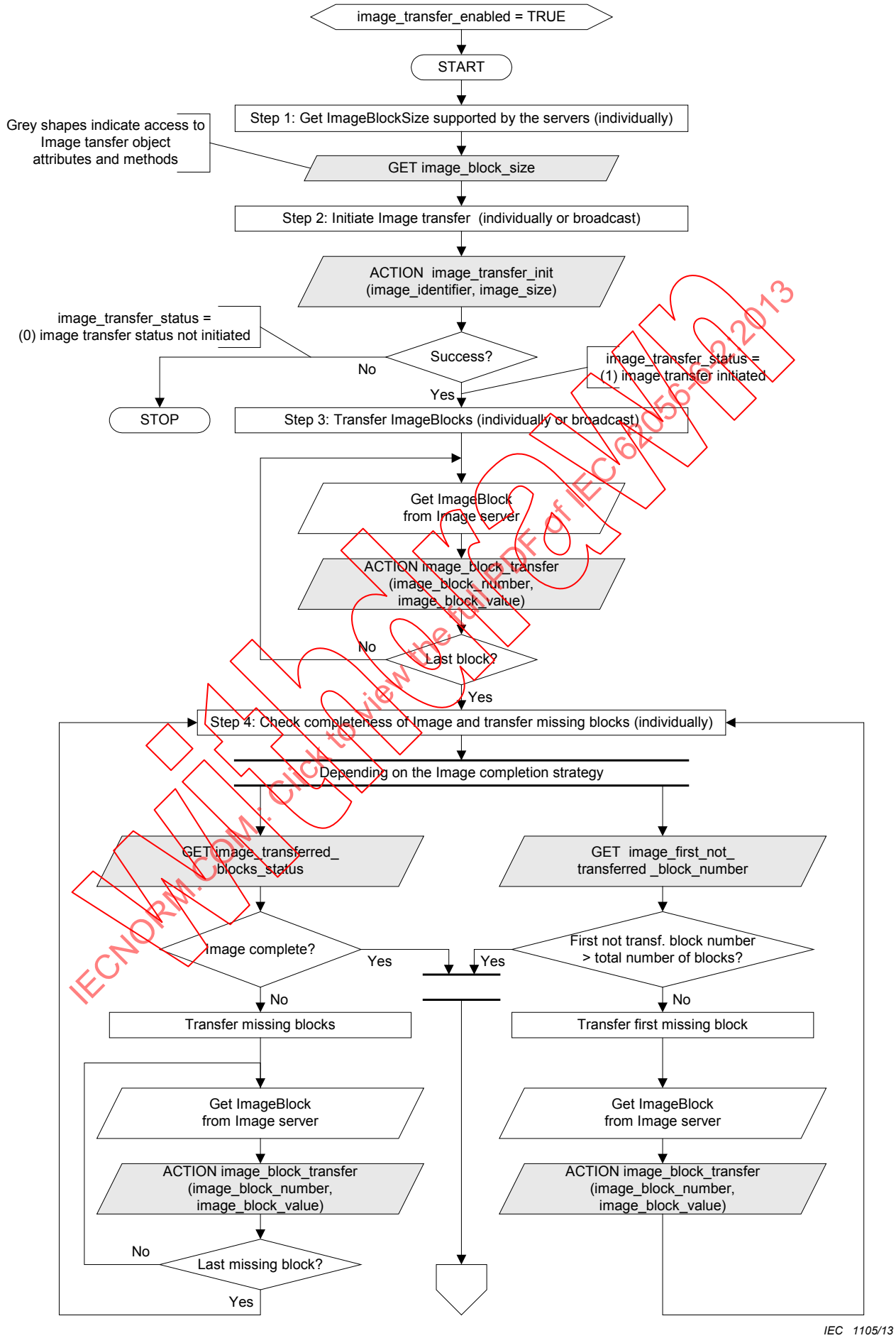
Étape 7: Activer l'Image: Dans la septième et dernière étape, l'Image est activée. Une Image transférée peut contenir une ou plusieurs Images à activer.

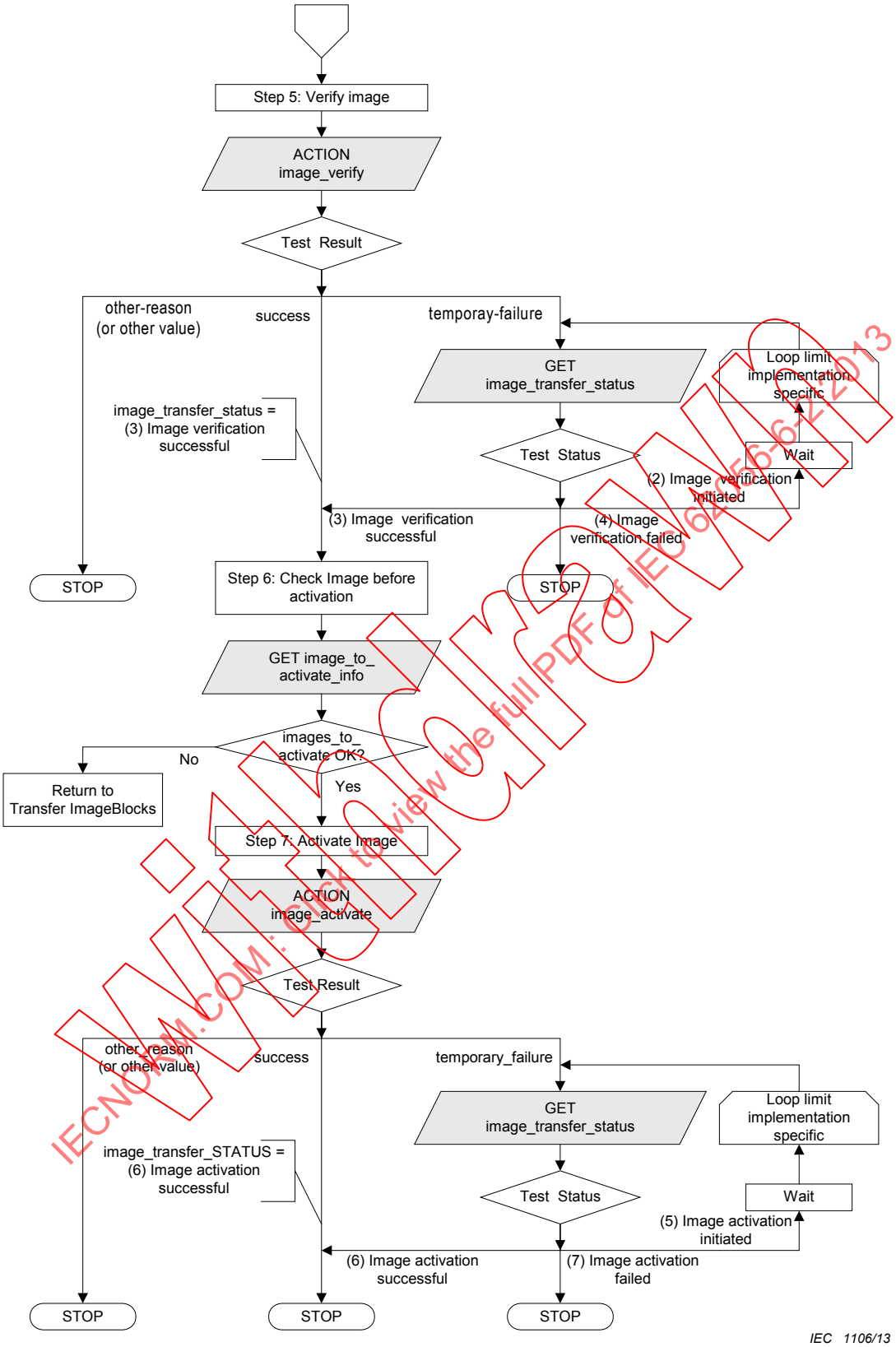
L'activation d'Image est réalisée en invoquant la méthode *image_activate*, soit par le client, soit par le serveur lui-même. Si l'activation est faite sans vérification préalable, une vérification est faite implicitement en tant que partie intégrante de l'activation. Le résultat peut être:

- success (succès), si l'activation a pu être achevée;
- temporary-failure (échec temporaire), si l'activation n'a pas pu être achevée;
- other-reason (autre raison), si l'activation a échoué.

NOTE 6 Pour les services xDLMS, les codes Action-Result / Data-Access-Result sont spécifiés dans la CEI 62056-5-3:—, Article 8.

Le résultat de l'activation de l'Image peut être vérifié par le client en récupérant la valeur de l'attribut *image_transfer_status*.





Légende

Anglais	Français
Grey shapes … methods	Les formes grises indiquent l'accès aux attributs et méthodes de l'objet transfert d'image
Step 1:…(individually)	Etape 1: Get ImageBlockSize pris en charge par les serveurs (individuellement)
Step 2: .. broadcast)	Etape 2: Déclencher le transfert d'image (individuellement ou en diffusion)
Image_transfer_status= (0) image transfer status not initiated	Image_transfer_status = (0) statut de transfert d'image non déclenché
Success ?	Succès
No	Non
Yes	Oui
Image_transfer_status= (1) image transfer initiated	Image_transfer_status= (1) transfert d'image déclenché
Step 3:…. Broadcast)	Étape 3: Transférer les ImageBlocks (individuellement ou en diffusion)
Get ImageBlock from Image server	Récupérer l'ImageBlock du serveur d'image
Last block ?	Dernier bloc ?
Step 4: Check … individually)	Etape 4: Vérifier la complétude de l'image et transférer les blocs manquants (individuellement)
Depending on …. Strategy	En fonction de la stratégie d'achèvement d'image
Image complete ?	Image complète ?
First not … blocks ?	Nombre de premiers blocs non transférés>nombre total de blocs ?
Transfer missing blocks	Transférer les blocs manquants
Transfert first missing block	Transférer le premier bloc manquant
Get ImageBlock from Image server	Récupérer ImageBlock du serveur d'Image
Last missing block ?	Dernier block manquant ?
Step 5: Verify image	Etape 5: Vérifier l'image
Loop limit implementation specific	Limite de boucle spécifique à la mise en œuvre
Image_transfer_status= (3) image verification successful	Image_transfer_status = (3) vérification d'image réussie
Test status	Résultat du test
Wait	Attendre
(2) image verification initiated	(2) vérification d'image lancée
(3) image verification successful	(3) vérification d'image réussie
(4) image verification failed	(4) vérification d'image échouée
Step 6: Check Image before activation	Étape 6: Vérifier l'Image avant activation
Return to Tranfer Imageblocks	Retourner à Transférer des blocs d'image
Activate Image	Activer l'image
Test Result	Résultat du test
Other-reason (or other value)	Other-reason (ou autre valeur)
Success	Success (succès)
Image_transfer_STATUS= (6) Image activation successful	Image_transfer_STATUS= (6) activation de l'Image réussie
Test status	Résultat du test
(5) image activation initiated	(5) activation d'image lancée
(6) image activation successful	(6) activation d'image réussie
(7) image activation failed	(7) activation d'image échouée

Figure 11 – Organigramme du processus de transfert d'Image

5.3.4.4 Définition de l'IC Image transfer

Image transfer (Transfert d'Image)		**0...n**	**class_id = 18, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. image_block_size	(static)	double-long-unsigned				x + 0x08
3. image_transferred_blocks_status	(dyn.)	bit-string				x + 0x10
4. image_first_not_transferred_block_number	(dyn.)	double-long-unsigned				x + 0x18
5. image_transfer_enabled	(static)	boolean				x + 0x20
6. image_transfer_status	(dyn.)	enumerated				x + 0x28
7. image_to_activate_info	(dyn.)	array				x + 0x30
Méthodes spécifiques		***m/o***				
1. image_transfer_initiate		m				x + 0x40
2. image_block_transfer		m				x + 0x48
3. image_verify		m				x + 0x50
4. image_activate		m				x + 0x58

Description d'attribut

logical_name

Identifie l'instance de l'objet "Image Transfer" (Transfert d'Image). Voir 6.2.26.

image_block_size

Contient l'ImageBlockSize, en octets, qui peut être géré par le serveur.

Il convient qu'il ne dépasse pas le ServerMaxReceivePduSize négocié.

image_transferred_blocks_status

Fournit les informations relatives à l'état du transfert de chaque ImageBlock. Chaque bit dans la chaîne de bits fournit des informations relatives à un ImageBlock pris séparément:

0 = Non transféré,
1 = Transféré

image_first_not_transferred_block_number

Fournit l'ImageBlockNumber du premier ImageBlock non transféré.

Si l'Image est complète, il convient que la valeur retournée soit supérieure au nombre de blocs calculé à partir de la taille de l'Image et de l'ImageBlockSize.

image_transfer_enabled

Contrôle l'autorisation du processus de transfert d'Image. Les méthodes ne peuvent être invoquées avec succès que si la valeur de cet attribut est TRUE.

boolean: FALSE = Désactivé,
TRUE = Activé

image_transfer_status

Contient l'état du processus de transfert d'Image.

enum:
(0) Transfert d'Image non déclenché,
(1) Transfert d'Image déclenché,
(2) Vérification d'Image déclenchée,
(3) Vérification d'Image réussie,
(4) Vérification d'Image échouée,
(5) Activation d'Image déclenchée,
(6) Activation d'Image réussie,
(7) Activation d'Image échouée

image_to_activate_info	Fournit les informations de l'Image ou des Images prête(s) à l'activation. Il est généré comme le résultat de la vérification de l'Image. Le client peut vérifier ces informations avant d'activer l'/les Image(s).
	array image_to_activate_info_element
	image_to_activate_info_element::= structure
	{
	image_size: double-long-unsigned,
	image_identification: octet-string,
	image_signature: octet-string
	}
	où
	• image_size est la taille de l'/des Image(s) à activer, exprimée en octets;
	• image_identification est l'identification de l'/des Image(s) à activer et peut contenir des informations telles que fabricant, type de dispositif, informations de version, etc.;
	• image_signature est la signature de l'/des Image(s) à activer.
	NOTE 1 Cet attribut contient des informations de l'/des Image(s) qui sera/seront activée(s). Dans le cas de l'électricité, les informations sur l'/les Image(s) actuellement active(s) sont détenues par l'objet 1.b.0.2.0.255, Numéro de version du programme de configuration (alias Version de firmware active). La signature de firmware active est détenue par l'objet 1.b.0.2.8.255. Pour d'autres supports, voir les articles pertinents.

Description de la méthode	

image_transfer_initiate (data)	Initialise le processus de transfert de l'Image.
	data::= structure
	{
	image_identifier: octet-string,
	image_size: double-long-unsigned
	}
	où:
	• image_identifier identifie l'Image à transférer;
	• image_size contient l'ImageSize, en octets.

image_block_transfer (data)	Transfère un bloc de l'Image vers le serveur.
	data::= structure
	{
	image_block_number: double-long-unsigned,
	image_block_value: octet-string
	}
	NOTE 2 Comme spécifié en 5.3.4.2, le premier ImageBlock envoyé est le bloc 0.

image_verify (data)	Vérifie l'intégrité de l'Image avant activation. data::= integer(0) Le résultat de l'invocation de cette méthode peut être "success", "temporary-failure" ou "other-reason". S'il n'est pas "success", le résultat de la vérification peut être connu en récupérant la valeur de l'attribut image_transfer_status. NOTE 3 Pour les services xDLMS, les codes Action-Result / Data-Access-Result sont spécifiés dans la CEI 62056-5-3:—, Article 8.

image_activate (data)	Active l'/les Image(s). data::= integer(0) Si l'Image transférée n'a pas été vérifiée au préalable, cela est fait comme partie intégrante de l'activation de l'Image. Le résultat de l'invocation de cette méthode peut être "success", "temporary-failure" ou "other-reason". S'il n'est pas "success", le résultat de l'activation peut être connu en récupérant la valeur de l'attribut image_transfer_status. NOTE 4 Pour les services xDLMS, les codes Action-Result / Data-Access-Result sont spécifiés dans la CEI 62056-5-3:—, Article 8.

5.3.5 Security setup (class_id: 64, version: 0)

Les instances de cette IC contiennent les informations nécessaires relatives à la politique de sécurité applicable et à la suite de sécurité utilisée au sein d'une AA particulière, entre deux systèmes identifiés respectivement par leur titre de système client et leur titre de système serveur. Elles contiennent aussi les méthodes pour augmenter le niveau de sécurité et transférer les clés globales. Voir aussi CEI 62056-5-3:—, Article 5.

Security setup (établissement de la sécurité)		**0…n**	**class_id = 64, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. security_policy	(static)	enum				x + 0x08
3. security_suite	(static)	enum				x + 0x10
4. client_system_title	(dyn.)	octet-string				x + 0x18
5. server_system_title	(static)	octet-string				x + 0x20
Méthodes spécifiques		***m/o***				
1. security_activate		o				x + 0x28
2. global_key_transfer		o				x + 0x30

Description d'attribut

logical_name	Identifie l'instance de l'objet “Security setup” (Établissement de la sécurité). Voir 6.2.25.
security_policy	Impose l'algorithme d'authentification et/ou de chiffrement fourni avec security_suite. enum: (0) aucune (1) tous les messages devant être authentifiés, (2) tous les messages devant être chiffrés, (3) tous les messages devant être authentifiés et chiffrés
security_suite	Spécifie l'algorithme d'authentification, de chiffrement et de transport de clé. enum: (0) AES-GCM-128 pour chiffrement authentifié et AES-128 pour l'enveloppement de clé

client_system_title	Détient le titre du système client (courant): • Dans l'environnement PLC S-FSK, l'initiateur actif envoie son titre de système en utilisant le protocole CIASE; NOTE 1 Il est également détenu par l'attribut active_initiator de l'objet S-FSK Active initiator; voir 5.8.5; • au cours de l'établissement d'AA confirmé ou non confirmé, il est transporté par le champ calling-AP-title de l'AARQ APDU; Si un titre du système client a déjà été envoyé au cours d'un processus d'enregistrement, comme dans le cas du profil PLC S-FSK, il convient que le titre du système client transporté par l'AARQ APDU soit le même. Autrement, l'AA doit être rejetée et des informations de diagnostic appropriées doivent être envoyées. • Dans une AA préétablie, il peut être écrit par le client avec l'aide d'un service SET / Write non sécurisé.
server_system_title	Transporte le titre du système serveur. • Dans l'environnement PLC S-FSK, le serveur envoie son titre de système au cours du processus de découverte, en utilisant le protocole CIASE; • au cours de l'établissement d'AA confirmé, il est transporté par le champ responding-AP-title de l'AARE APDU. Cet attribut doit être en lecture seulement.
Description de la méthode	
security_activate (data)	Active et renforce la politique de sécurité: enum: (0) aucune (1) tous les messages devant être authentifiés, (2) tous les messages devant être chiffrés, (3) tous les messages devant être authentifiés et chiffrés NOTE 2 La politique de sécurité peut seulement être renforcée.

global_key_transfer (data)	Met à jour une ou plusieurs clés globales.
	Le paramètre *data* comprend des données de clé enveloppée. Les données de clé comprennent les identificateurs de clés et les clés elles-mêmes.
	array key_data
	key_data::= structure
	{
	key_id: enum (0) clé globale de chiffrage unicast,
	(1) clé globale de chiffrage en diffusion,
	(2) clé d'authentification
	key_wrapped: octet-string
	}
	L'algorithme d'enveloppement de clé est tel que spécifié par la suite de sécurité. La KEK est la clé maîtresse.

5.4 Classes d'interfaces pour commande à limite temporelle et événementielle

5.4.1 Clock (class_id: 8, version: 0)

Cette IC modélise l'horloge, gérant toutes les informations relatives à la date et à l'heure, y compris les décalages de l'heure locale par rapport à une référence temporelle généralisée (temps universel coordonné, TUC), en raison des fuseaux horaires et des schémas de changement d'horaire légaux. L'IC propose également diverses méthodes pour régler l'horloge.

Les informations de date comprennent les éléments year (année), month (mois), day of month (jour dans le mois) et day of week (jour dans la semaine). Les informations d'heure comprennent les éléments hour (heure), minutes, seconds (secondes), hundredths of seconds (centièmes de seconde) et le décalage de l'heure locale par rapport au temps TUC. La fonction heure (été/hiver) de changement légal modifie le décalage de l'heure locale par rapport au TUC en fonction des attributs. Le point initial et le point final de la fonction en question sont normalement fixés une seule fois. Un algorithme interne calcule le point réel de commutation en fonction de ces valeurs fixées. Voir Figure 12.

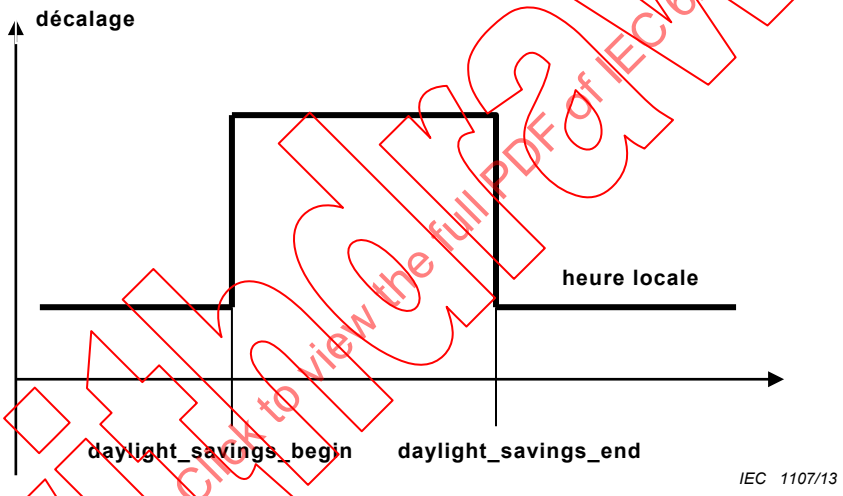


Figure 12 – Concept de temps généralisé

Clock (horloge)		**0...n**	**class_id = 8, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. time	(dyn.)	octet-string				x + 0x08
3. time_zone	(static)	long				x + 0x10
4. status	(dyn.)	unsigned				x + 0x18
5. daylight_savings_begin	(static)	octet-string				x + 0x20
6. daylight_savings_end	(static)	octet-string				x + 0x28
7. daylight_savings_deviation	(static)	integer				x + 0x30
8. daylight_savings_enabled	(static)	boolean				x + 0x38
9. clock_base	(static)	enum				x + 0x40
Méthodes spécifiques		***m/o***				
1. adjust_to_quarter (data)		o				x + 0x60
2. adjust_to_measuring_period (data)		o				x + 0x68
3. adjust_to_minute (data)		o				x + 0x70

Clock (horloge)	**0...n**	**class_id = 8, version = 0**	
4. adjust_to_preset_time (data)	o		x + 0x78
5. preset_adjusting_time (data)	o		x + 0x80
6. shift_time (data)	o		x + 0x88

Description d'attribut

logical_name	Identifie l'instance de l'objet “Clock” (Horloge). Voir 6.2.4.
time	Contient les date et heure locales du compteur, son écart par rapport au TUC et le statut. Voir 4.6.1. Lorsque cette valeur est établie, seuls les champs spécifiés du date_time sont modifiés. Par exemple, pour régler la date sans modifier l'heure, tous les octets relatifs à l'heure dans date_time doivent être mis à “not specified” (non spécifié). Le clock_status doit toujours être mis lors de l'écriture de l'heure. octet-string, formaté comme indiqué en 4.6.1 pour date_time
time_zone	L'écart de l'heure locale normale par rapport au TUC en minutes.
status	L’état est égal à l’état lu dans time. Voir 4.6.1. unsigned (non signé), formaté tel que précisé en 4.6.1 pour clock_status
daylight_savings_ begin	Définit les date et heure locales de basculement lorsque l'heure locale a été décalée par rapport à l'heure normale. Pour des définitions génériques, il est permis d'utiliser des caractères jokers. octet-string, formaté comme indiqué en 4.6.1 pour date_time
daylight_savings_ end	Voir ci-dessus. octet-string, formaté comme indiqué en 4.6.1 pour date_time
daylight_savings_ deviation	Contient le nombre de minutes dont l'écart d'heure généralisée doit être corrigé au daylight_savings_begin (moment du début d'heure (été/hiver)). integer: Plage d'écart jusqu'à ± 120 min
daylight_savings_ enabled	boolean: TRUE = DST enabled (heure hiver/été activée) FALSE = DST disabled (heure hiver/été désactivée)
clock_base	Définit la provenance des informations de temporisation de base. enum: (0) non définie, (1) quartz interne, (2) fréquence secteur de 50 Hz,

(3) fréquence secteur de 60 Hz,
(4) GPS (système de positionnement global)),
(5) radiocommandée

Description de la méthode	
adjust_to_quarter (data)	Règle l'heure du compteur sur le plus proche (+/-) quart d'heure (*:00, *:15, *:30, *:45). data::= integer(0)
adjust_to_measuring_period (data)	Règle l'heure du compteur sur le plus proche (+/-) point de départ d'une période de mesure. data::= integer(0)
adjust_to_minute (data)	Règle l'heure du compteur sur la minute la plus proche. Si second_counter < 30 s, le second_counter est donc mis à 0. Si second_counter ≥ 30 s, donc second_counter est mis à 0 et les valeurs de minute_counter et toutes les valeurs d'horloge dépendantes sont incrémentées si besoin est. data::= integer(0)
adjust_to_preset_time (data)	Cette méthode est utilisée conjointement à la méthode preset_adjusting_time. Si l'heure du compteur s'inscrit entre validity_interval_start et validity_interval_end, l'heure est mise à preset_time. data::= integer(0)

preset_adjusting_time (data)	Prérègle l'heure sur une nouvelle valeur (preset_time) et définit un validity_interval dans lequel la nouvelle heure peut être activée.
	
	data::= structure
	{
	preset_time: octet-string,
	validity_interval_start: octet-string,
	validity_interval_end: octet-string
	}
	tous les octet-string, formatés comme indiqué en 4.6.1 pour *date_time*

shift_time (data)	Décale l'heure de *n* (-900 <= *n* <= 900) s. data::= long

5.4.2 Script table (class_id: 9, version: 0)

Cette IC permet de modéliser le déclenchement d'une série d'actions en exécutant des scripts par la méthode execute (data).

Les objets "Script table" contiennent un tableau d'entrées de scripts. Chaque entrée consiste en un *script identifier* (identificateur de script) et une série de *action specifications*

(spécifications d'action). Une spécification d'action active une méthode ou modifie un attribut d'un objet COSEM au sein d'un dispositif logique.

Un certain script peut être activé par d'autres objets COSEM au sein du même dispositif logique ou depuis l'extérieur.

Si deux scripts doivent être exécutés dans la même instance temporelle, c'est celui qui a l'indice le plus petit qui est exécuté le premier.

Script table (tableau de scripts)	**0...n**	**class_id = 9, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. scripts (static)	array				x + 0x08
Méthodes spécifiques	***m/o***				
1. execute(data)	m				x + 0x20

Description d'attribut

logical_name

Identifie l'instance de l'objet "Script table " (Tableau de scripts). Voir 6.2.6.

scripts

Spécifie les différents scripts, à savoir la liste d'actions.

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

Le script_identifier 0 est réservé. S'il est spécifié avec une méthode d'exécution, il se traduit en un script vide (aucune action à accomplir).

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

où

service_id: définit l'action à appliquer à l'objet référencé.

(1) écrire un attribut,

(2) exécuter une méthode spécifique

index: définit (avec service_id 1) quel attribut de l'objet sélectionné est affecté; ou (avec service_id 2) quelle méthode spécifique est à exécuter.

Le premier attribut (logical_name) a l'indice 1, la première méthode spécifique a l'indice 1 également.

NOTE 1 L'action_specification se limite à activer des méthodes qui ne produisent pas de réponse (du serveur au client).

NOTE 2 Une spécification d'action "dummy" (fictive) avec tous les éléments ayant la valeur 0 signifie que l'action n'est pas configurée.

Description de la méthode

execute (data)	Exécute le script spécifié dans les données de paramètre. data::= long-unsigned Si data concorde avec l'un des script_identifier dans le tableau de scripts, l'action_specification correspondante est exécutée.

5.4.3 Schedule (class_id: 10, version: 0)

Cette IC, avec l'IC “Special days” (jours spéciaux), permet de modéliser des activités qui dépendent de l'heure et de la date au sein d'un dispositif. Les tableaux suivants fournissent une vue générale et montrent les interactions entre les deux IC.

Tableau 5 – Schedule (programme)

Index (Indice)	enable (active)	action (script)	Switch_time	validity_window	exec_weekdays							exec_specdays					plage de dates	
					Mo (Lu)	Tu (ma)	We (me)	Th (jeudi)	Fr (vendredi)	Sa	Su (Dimanche)	S1	S2	...	S8	S9	begin_date	end_date
120	Oui	xxxx:yy	06:00	0xFFFF	*x*	*x*	*x*	*x*	*x*	*x*							xx-04-01	xx-09-30
121	Oui	xxxx:yy	22:00	15	*x*	*x*	*x*	*x*	*x*								xx-04-01	xx-09-30
122	Oui	xxxx:yy	12:00	0						*x*							xx-04-01	xx-09-30
200	Non	xxxx:yy	06:30		*x*	*x*	*x*	*x*	*x*	*x*							xx-04-01	xx-09-30
201	Non	xxxx:yy	21:30		*x*	*x*	*x*	*x*	*x*								xx-04-01	xx-09-30
202	Non	xxxx:yy	11:00							*x*							xx-04-01	xx-09-30

Tableau 6 – Special days table (Tableau de jours spéciaux)

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

Schedule		**0...n**	**class_id = 10, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. entries	(static)	array				x + 0x08
Méthodes spécifiques		***m/o***				
1. enable/disable (data)		O				x + 0x20
2. insert (data)		O				x + 0x28
3. delete (data)		O				x + 0x30

Description d'attribut

logical_name	Identifie l'instance de l'objet “Schedule” (Programme). Voir 6.2.8.
entries	Spécifie les scripts devant être exécutés à des instants donnés. Un seul script peut être exécuté par entrée. 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
}
```

où:

- script_logical_name: définit le nom logique de l'objet "Script table";
- script_selector: définit le script_identifier du script à exécuter;
- switch_time accepte les caractères jokers pour définir les entrées répétitives. Le format de l'octet-string suit les règles établies en 4.6.1 pour *time*;
- validity_window définit une période, en minutes, dans laquelle une entrée doit être traitée après une coupure secteur . (temps entre le switch_time défini et le power_up effectif) 0xFFFF: le script doit être traité à tout instant;
- exec_weekdays définit les jours de la semaine pendant lesquels l'entrée est valide;
- exec_specdays établit le lien à l'IC "Special days table", day_id;
- begin_date et end_date définissent la date dans laquelle l'entrée est valide (les jokers ont permis). Le format suit les règles établies en 4.6.1 pour *date*.

Description de la méthode	
enable/disable (data)	Met le bit désactivé des entrées de la plage A à true (vrai) puis active les entrées de la plage B.

```
data::= structure
{
    firstIndexA: long-unsigned,
    lastIndexA: long-unsigned,
    firstIndexB: long-unsigned,
    lastIndexB: long-unsigned
}
```

firstIndexA	premier indice de la plage qui est désactivée
lastIndexA	dernier indice de la plage qui est désactivée
firstIndexB	premier indice de la plage qui est activée
lastIndexB	dernier indice de la plage qui est activée
firstIndexA/B < lastIndexA/B:	toutes les entrées de la plage A/B sont désactivées/activées
firstIndexA/B == lastIndexA/B:	une entrée est désactivée/activée

	firstIndexA/B > lastIndexA/B:	rien n'est désactivé/activé
	firstIndexA/B et lastIndexA/B > 9999:	aucune entrée n'est désactivée/activée
insert (data)	Insère une nouvelle entrée dans le tableau. Si l'indice de l'entrée existe déjà, l'entrée existante est écrasée par la nouvelle entrée. entry (entrée): schedule_table_entry data (donnée): correspondant à entrée	

delete (data)	Efface une plage d'entrées dans le tableau.
	
	data::= structure
	{
	firstIndex: long-unsigned,
	lastIndex: long-unsigned
	}	

	firstIndex	premier indice de la plage qui est effacée
	lastIndex	dernier indice de la plage qui est effacée
	firstIndex < lastIndex:	toutes les entrées de la plage A/B sont effacées,
	firstIndex::= lastIndex:	une entrée est effacée,
	firstIndex > lastIndex:	rien n'est effacé

Remarques concernant les "incohérences" dans les entrées du tableau:

- S'il convient d'exécuter plusieurs fois le même script à une instance de time spécifique, il n'est exécuté qu'une seule fois.
- S'il convient d'exécuter des scripts différents dans la même instance de time, l'ordre d'exécution est selon l'indice. Le script à l'indice le plus bas est exécuté le premier.

- **Récupération après une coupure secteur**

Après une coupure secteur, le programme tout entier est traité pour exécuter tous les scripts nécessaires qui seraient perdus pendant une coupure secteur. Pour cela, les entrées qui n'ont pas été exécutées pendant la coupure secteur doivent être détectées. En fonction de l'attribut fenêtre de validité, elles sont exécutées dans l'ordre correct (selon lequel elles auraient été exécutées en fonctionnement normal).

- **Gestion des changements de temps**

Il existe quatre "actions" différentes de changements de temps:

a) réglage de l'heure en avance (en écrivant dans l'attribut "time" de l'objet "Clock");
b) réglage de l'heure en retard (en écrivant dans l'attribut "time" de l'objet "Clock");
c) synchronisation du temps (en utilisant la méthode "adjust_to_quarter" de l'objet "Clock");
d) action d'économie d'énergie (heure d'été/hiver).

Toutes ces quatre actions nécessitent une gestion différente exécutée par le programme en interaction avec l'activité de réglage de l'heure.

- **Réglage de l'heure en avance**

Il est géré de la même façon qu'une coupure secteur. Toutes les entrées manquées sont exécutées en fonction de l'attribut fenêtre de validité. Un réglage de temps court (défini et spécifique au fabricant) peut être géré comme synchronisation du temps.

- **Réglage de l'heure en retard**

Il se traduit par une répétition des entrées qui sont activées pendant le temps répété. Un réglage de temps court (défini et spécifique au fabricant) peut être géré comme synchronisation du temps.

- **Synchronisation**

La synchronisation du temps est utilisée pour corriger de petits écarts entre l'horloge centrale (maîtresse) et l'horloge locale. L'algorithme est spécifique au fabricant. Il doit garantir qu'aucune entrée du programme ne se perd ou n'est exécuté deux fois. L'attribut fenêtre de validité n'a aucun effet, car toutes les entrées doivent être exécutées en fonctionnement normal.

- **Heure (été/hiver) de changement d'horaire légal**

Si l'horloge est mise en avance, tous les scripts qui s'inscrivent dans l'intervalle d'avance (et seraient donc perdus) sont exécutés. Si l'horloge est remise en arrière, la réexécution des scripts qui s'inscrivent dans l'intervalle de retard est arrêtée.

5.4.4 Special days table (class_id: 11, version: 0)

Cette IC permet de définir des dates spéciales. À ces dates, un comportement de commutation spécial prévaut sur le comportement normal. L'IC fonctionne conjointement à la classe "Schedule" ou "Activity calendar". L'élément de donnée de liaison est *day_id*.

Special days table (tableau de jours spéciaux)	**0...1**	**class_id = 11, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. entries (static)	array				x + 0x08
Méthodes spécifiques	***m/o***				
1. insert (data)	o				x + 0x10
2. delete (data)	o				x + 0x18

Description d'attribut

logical_name Identifie l'instance de l'objet “Special days table” (Tableau de jours spéciaux). Voir 6.2.7.

entries Spécifie un identificateur de jour spécial pour une date donnée. La date peut comporter des jokers pour les jours spéciaux qui se répètent, comme Noël.

array spec_day_entry

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

- le formatage de specialday_date suit les règles établies en 4.6.1 pour *date*;
- la plage du day_id doit concorder avec la longueur de la chaîne binaire exec_specdays dans l'objet connexe de l'IC “Schedule”.

Description de la méthode	
insert (data)	Insère une nouvelle entrée dans le tableau. entry::= spec_day_entry Si un jour spécial ayant le même indice ou la même date qu'un jour déjà défini est inséré, l'ancienne entrée sera écrasée.
delete (data)	Efface une entrée dans le tableau. index (indice) Indice de l'entrée qui doit être supprimée. data::= long-unsigned

5.4.5 Activity calendar (class_id: 20, version: 0)

Cette IC permet de modéliser la gestion de diverses structures de tarification dans le compteur. L'IC fournit une liste d'actions programmées, suivant le format classique des programmes basés sur un calendrier en définissant des saisons, des semaines, etc.

Un objet “Activity calendar” peut coexister avec l’objet plus général “Schedule” et peut même se superposer à celui-ci. Si des actions dans un objet “Schedule” sont programmées pour la même heure d'activation que dans un objet “Activity calendar”, les actions déclenchées par l'objet “Schedule” sont exécutées les premières.

Après une coupure secteur, la “last action” (dernière action) manquée issue de l'objet “Activity calendar” est exécutée (différée). Le but en est d'assurer une tarification correcte après une mise sous tension. Si un objet “Schedule” est présent, la "dernière action" manquée de l'objet “Activity calendar” doit être exécutée à l'heure correcte dans la séquence d'actions demandées par l'objet “Schedule”.

L'objet “Activity calendar” définit l'activation de certains scripts, qui peuvent accomplir différentes activités à l'intérieur du dispositif logique. L'interface à l'IC “Script table” est la même que dans le cas de l'IC “Schedule” (voir 5.4.3).

Si une instance de l'IC “Special days table” (voir 5.4.4) est disponible, les entrées pertinentes prévalent sur la sélection d'un profil de jour commandée par l'objet “Activity calendar”. Le profil de jour référencé dans le “Special days table” active le day_schedule du day_profile_table dans l'objet “Activity calendar” par référencement par le biais du day_id.

Activity calendar (calendrier d'activités)		0...n	class_id = 20, version = 0			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. calendar_name_active	(static)	octet-string				x + 0x08
3. season_profile_active	(static)	array				x + 0x10
4. week_profile_table_active	(static)	array				x + 0x18
5. day_profile_table_active	(static)	array				x + 0x20
6. calendar_name_passive	(static)	octet-string				x + 0x28
7. season_profile_passive	(static)	array				x + 0x30
8. week_profile_table_passive	(static)	array				x + 0x38
9. day_profile_table_passive	(static)	array				x + 0x40
10. activate_passive_calendar_time	(static)	octet-string				x + 0x48
Méthodes spécifiques		***m/o***				
1. activate_passive_calendar (data)		o				x + 0x50

Description d'attribut

Les attributs appelés ..._active sont actuellement actifs, ceux appelés ..._passive seront activés par la méthode spécifique activate_passive_calendar.

logical_name Identifie l'instance de l'objet "Activity calendar" (Calendrier d'activités). Voir 6.2.9.

calendar_name Contient typiquement un identificateur, qui est descriptif du jeu de scripts activés par l'objet.

season_profile Contient une liste de saisons définies par leur date de début et un week_profile spécifique devant être exécuté. La liste est triée selon season_start (par ordre croissant):

```
array season

season::= structure
{
        season_profile_name:    octet-string,
        season_start:           octet-string,
        week_name:              octet-string
}
```

où:

- season_profile_name est un nom défini par l'utilisateur qui identifie le season_profile courant;
- season_start définit l'heure de début de la saison, formatée comme indiqué en 4.6.1 pour *date_time*. Les jokers sont autorisés dans season_start. Si tous les champs sont des jokers, la saison ne commencera jamais. L'utilisation de jokers exige un soin particulier pour éviter toute paramétrisation conflictuelle, c'est-à-dire éviter que deux saisons différentes aient le même season_start.

NOTE La saison en cours est terminée par le season_start de la saison suivante.

- week_name définit le week_profile actif dans cette saison.

week_profile_table Contient un tableau de week_profiles à utiliser dans différentes saisons. Pour chaque week_profile, le day_profile pour chaque jour d'une semaine est identifié.

	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
	où:
	• week_profile_name est un nom défini par l'utilisateur qui identifie le week_profile courant;
	• Monday (lundi) définit le day_profile valide le lundi;
	• …
	• Sunday (dimanche) définit le day_profile valide le dimanche.

day_profile_table	Contient un tableau de day_profiles, identifiés par leur day_id. Pour chaque day_profile, une liste d'actions programmées est définie par un script à exécuter et l'heure d'activation (start_time) correspondante. La liste est triée selon start_time.
	
	array day_profile
	day_profile::= structure
	{
	day_id: unsigned,
	day_schedule: array day_profile_action
	}
	
	day_profile_action::= structure
	{
	start_time: octet-string,
	script_logical_name: octet-string,
	script_selector: long-unsigned
	}
	où:
	• day_id est un identificateur défini par l'utilisateur, qui identifie le day_profile courant;
	• start_time: définit l'heure à laquelle le script doit être exécuté (aucun joker); Le format suit les règles établies en 4.6.1 pour *time*;
	• script_logical_name: définit le nom logique de l'objet “Script table”;
	• script_selector: définit le script_identifier du script à exécuter.

activate_passive_ calendar_time	Définit l'heure à laquelle l'objet lui-même appelle la méthode spécifique activate_passive_calendar. Une définition avec une notation "not specified" (c'est-à-dire: non spécifié) dans tous les champs de l'attribut désactivera cet automatisme. Une notation "not specified" partielle, juste dans quelques champs de la date et de l'heure, n'est pas autorisée. octet-string, formaté comme indiqué en 4.6.1 pour *date_time*
Description de la méthode	

activate_passive_ calendar(data)	Cette méthode copie tous les attributs appelés ..._passive dans les attributs correspondants appelés ..._active. data::= integer(0)

5.4.6 Register monitor (class_id: 21, version: 0)

Cette IC permet de modéliser la fonction de surveillance de valeurs modélisées par les objets “Data”, “Register”, “Extended register” ou “Demand register”. Elle permet de spécifier des seuils, la valeur surveillée et un jeu de scripts (voir 5.4.2) qui sont exécutés lorsque la valeur surveillée franchit un seuil.

L'IC “Register monitor” requiert une instanciation de l'IC “Script table” dans le même dispositif logique.

Register monitor (Moniteur de registres)		**0...n**	**class_id = 21, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. thresholds	(static)	array				x + 0x08
3. monitored_value	(static)	value_definition				x + 0x10
4. actions	(static)	array			0	x + 0x18
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name	Identifie l'instance de l'objet “Register monitor” (Observateur de registres). Voir 6.2.12 et 6.3.10.
thresholds (seuils)	Fournit les valeurs de seuil auxquelles l'attribut du registre référencé est comparé. array threshold threshold: Le seuil est du même type que l'attribut surveillé de l'objet référencé.

monitored_value	Définit quel attribut d'un objet est à surveiller. Seules sont autorisées les valeurs ayant des types de donnée simples.
	
	value_definition::= structure
	
	{
	class_id: long-unsigned,
	logical_name: octet-string,
	attribute_index: integer
	}

actions	Définit les scripts à exécuter lorsque l'attribut surveillé de l'objet référencé franchit le seuil correspondant. L'attribut “actions” a exactement le même nombre d'éléments que l'attribut “thresholds”. L'ordre de classement des action_items correspond à l'ordre de classement des thresholds (seuils, voir ci-dessus). array action_set

```
action_set::=        structure
{
        action_up:  action_item,
        action_down:        action_item
}
```

où:

- action_up définit l'action lorsque la valeur d'attribut du registre surveillé franchit le seuil dans le sens montant;
- action_down définit l'action lorsque la valeur d'attribut du registre surveillé franchit le seuil dans le sens descendant;

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

5.4.7 Single action schedule (class_id: 22, version: 0)

Cette IC permet de modéliser l'exécution d'actions périodiques au sein d'un compteur. Ces actions ne sont pas nécessairement liées à la tarification (voir "Activity calendar" ou "Schedule").

Single action schedule (programme d'action unique)		**0...n**	**class_id = 22, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. executed_script	(static)	script				x + 0x08
3. type	(static)	enum				x + 0x10
4. execution_time	(static)	array				x + 0x18
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name

Identifie l'instance d'objet "Single action schedule" (programme d'action unique). Voir 6.2.11.

executed_script

Contient le nom logique de l'objet "Script table" et le sélecteur de script du script à exécuter.

```
script::= structure
{
        script_logical_name:        octet-string,
        script_selector:            long-unsigned
}
```

Script_logical_name et script_selector définissent le script à exécuter.

type	enum:	(1) taille de execution_time = 1; joker autorisé dans *date*, (2) taille de execution_time = *n*; toutes les valeurs de *time* sont les mêmes, les jokers sont interdits dans *date*, (3) taille de execution_time = *n*; toutes les valeurs de *time* sont les mêmes, les jokers sont autorisés dans *date*, (4) taille de execution_time = *n*; les valeurs de *time* peuvent être différentes, les jokers sont interdits dans *date*, (5) taille de execution_time = *n*; les valeurs de *time* peuvent être différentes, les jokers sont autorisés dans *date*,

execution_time	Spécifie l'heure et la date auxquelles le script est exécuté.
	array execution_time_date
	execution_time_date::= structure
	{
	time: octet-string,
	date: octet-string
	}
	les deux octet-strings contiennent *time* et *date*, dans cet ordre; *time* et *date* sont formatées comme défini en 4.6.1.
	Les WILDCARDS (jokers) dans *time* ne sont pas autorisés; les secondes et les centièmes de seconde doivent être zéro.	

5.4.8 Disconnect control (class_id: 70, version: 0)

Les instances de l'IC "Disconnect control" (commande de déconnexion) gèrent une unité de déconnexion interne ou externe du compteur (par exemple: disjoncteur d'électricité, robinet de gaz) afin de brancher ou débrancher - partiellement ou totalement - les locaux du consommateur à/de l'alimentation. Le diagramme d'états et les possibles transitions d'état sont montrés à la Figure 13.

Une déconnexion et une reconnexion peuvent être demandées:

- À distance, via une voie de communications: remote_disconnect, remote_reconnect;
- Manuellement, à l'aide d'un bouton-poussoir, par exemple: manual_disconnect, manual_reconnect;
- Localement, par une fonction du compteur, par exemple limiteur, prépaiement: local_disconnect, local_reconnect.

Les états et les transitions d'état de l'IC "Disconnect control" sont montrés au Tableau 7. Les transitions d'état possibles dépendent du mode de commande. L'objet "Disconnect control" ne comporte pas de mémoire, à savoir toute commande est exécutée immédiatement.

Afin de définir le comportement de l'objet "commande de déconnexion" pour chaque déclencheur, le mode de commande doit être configuré.

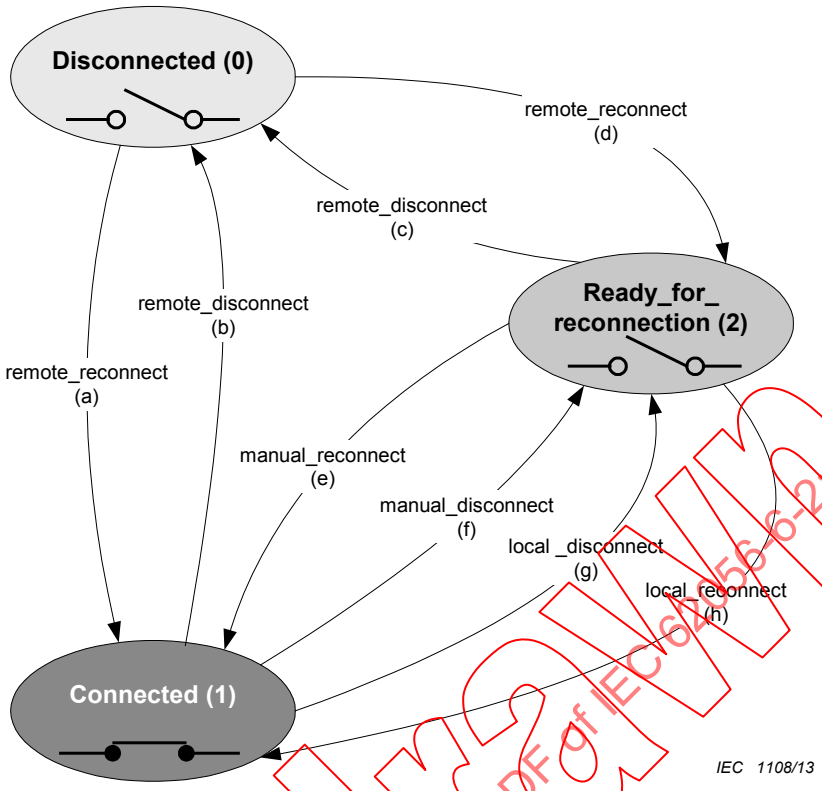


Figure 13 – Diagramme d'états de l'IC "Disconnect control"

Tableau 7 – IC "Disconnect control" – états et transitions d'état

États		
Numéro d'état	**Nom d'état**	**Description d'état**
0	Disconnected (déconnecté)	L'output_state est mis à FAUX et le consommateur est déconnecté.
1	Connected (connecté)	L'output_state est mis à VRAI et le consommateur est connecté.
2	Ready_for_reconnection	L'output_state est mis à FALSE et le consommateur est déconnecté.
Transitions d'état		
Transition	**Nom de transition**	**Description d'état**
a	remote_reconnect	Déplace l'objet "Disconnect control" de l'état Disconnected (0) directement à l'état Connected (1) sans intervention manuelle.
b	remote_disconnect	Déplace l'objet "Disconnect control" de l'état Connected (1) à l'état Disconnected (0).
c	remote_disconnect	Déplace l'objet "Disconnect control" de l'état Ready_for_reconnection (2) à l'état Disconnected (0).
d	remote_reconnect	Déplace l'objet "Disconnect control" de l'état Disconnected (0) à l'état Ready_for_reconnection (2). À partir de cet état, il est possible de passer à l'état Connected (2) via la transition manual_reconnect (e) ou la transition local_reconnect (h).
e	manual_reconnect	Déplace l'objet "Disconnect control" de l'état Ready_for_connection (2) à l'état Connected (1).
f	manual_disconnect	Déplace l'objet "Disconnect control" de l'état Connected (1) à l'état Ready_for_connection (2). À partir de cet état, il est possible de revenir à l'état Connected (2) via la transition manual_reconnect (e) ou la transition local_reconnect (h).
g	local_disconnect	Déplace l'objet "Disconnect control" de l'état Connected (1) à l'état Ready_for_connection (2). À partir de cet état, il est possible de revenir à l'état Connected (2) via la transition manual_reconnect (e) ou la transition local_reconnect (h). NOTE 1 Les transitions f) et g) sont essentiellement les mêmes mais leur déclencheur est différent.
h	local_reconnect	Déplace l'objet "Disconnect control" de l'état Ready_for_connection (2) à l'état Connected (1). NOTE 2 Les transitions e) et h) sont essentiellement les mêmes mais leur déclencheur est différent.

Disconnect control (commande de déconnexion)	**0…n**	**class_id = 70, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	***Nom court***
1. logical_name (static)	octet-string				x
2. output_state (dyn.)	boolean				x + 0x08
3. control_state (dyn.)	enum				x + 0x10
4. control_mode (static)	enum				x + 0x18
Méthodes spécifiques	**m/o**				
1. remote_disconnect()	m				x + 0x20
2. remote_reconnect()	m				x + 0x28

Description d'attribut

logical_name	Identifie l'instance de l'objet “Disconnect control” (commande de déconnexion). Voir 6.2.18 et 6.2.32.

output_state	Montre l'état physique réel de l'unité de déconnexion, c'est-à-dire si un disjoncteur d'électricité ou un robinet de gaz est ouvert ou fermé. boolean: TRUE (BOOLEAN 1) = Connected (c'est-à-dire connecté); FALSE (BOOLEAN 0) = Disconnected (c'est-à-dire déconnecté). NOTE 1 En comptage d'électricité, l'alimentation est connectée lorsque le dispositif déconnecteur est fermé. NOTE 2 En comptage de gaz et d'eau, l'alimentation est connectée lorsque le robinet est ouvert.
control_state	Montre l'état interne de l'objet de commande de déconnexion. enum: (0) Disconnected (déconnecté), (1) Connected (connecté), (2) Ready_for_reconnection (prêt pour la reconnexion)
control_mode	Configure le comportement de l'objet commande de déconnexion pour tous les déclencheurs, c'est-à-dire les transitions d'état possibles. enum: (0) None (aucune). L'objet commande de déconnexion est toujours dans l'état 'connected' (connecté), (1) Disconnection (déconnexion): Distante (b, c), manuelle (f), locale (g) Reconnection (reconnexion): Distante (d), manuelle (e), (2) Disconnection (déconnexion): Distante (b, c), manuelle (f), locale (g) Reconnection (reconnexion): Distante (a), manuelle (e), (3) Disconnection (déconnexion): Distante (b, c), manuelle (-), locale (g) Reconnection (reconnexion): Distante (d), manuelle (e), (4) Disconnection (déconnexion): Distante (b, c), manuelle (-), locale (g) Reconnection (reconnexion): Distante (a), manuelle (e) (5) Disconnection (déconnexion): Distante (b, c), manuelle (f), locale (g) Reconnection (reconnexion): Distante (d), manuelle (e), locale (h), (6) Disconnection (déconnexion): Distante (b, c), manuelle (-), locale (g) Reconnection (reconnexion): Distante (d), manuelle (e), locale (h) NOTE 3 La déconnexion locale est toujours possible, sauf si le déclencheur correspondant est inhibé.

Description de la méthode

remote_disconnect (data)	Force l'objet contrôle de déconnexion à l'état 'disconnected' si la déconnexion distante est activée (control mode > 0). data::= integer(0)

remote_reconnect (data)	Force l'objet contrôle de déconnexion à l'état 'ready_for_reconnection' si une reconnexion distante directe est désactivée (control_mode = 1, 3, 5, 6). Force l'objet contrôle de déconnexion à l'état 'connected' si une reconnexion distante directe est activée (control_mode = 2, 4). data::= integer(0)

5.4.9 Limiter (class_id: 71, version: 0)

Les instances de l'IC "Limiter" permettent de définir un jeu d'actions qui sont exécutées lorsque la valeur d'un attribut *value* d'un objet surveillé "Data", "Register", "Extended Register", "Demand Register", etc. franchit la valeur seuil pendant au moins la durée minimale.

La valeur seuil peut être un seuil normal ou un seuil d'urgence. Le *seuil d'urgence* est activé via le *profil d'urgence* défini par l'*id de profil d'urgence*, le *temps d'activation d'urgence* et la *durée d'urgence*. L'élément *emergency profile id* (id de profil d'urgence) est mis en concordance avec un *emergency profile group id* (id de groupe de profils d'urgence): ce mécanisme permet l'activation du seuil d'urgence uniquement pour un groupe spécifique d'urgences.

Limiter (limiteur)		**0...n**	**class_id = 71, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	***Nom court***
1. logical_name	(static)	octet-string				x
2. monitored_value	(static)	value_definition				x + 0x08
3. threshold_active	(dyn.)	threshold				x + 0x10
4. threshold_normal	(static)	threshold				x + 0x18
5. threshold_emergency	(static)	threshold				x + 0x20
6. min_over_threshold_duration	(static)	double-long-unsigned				x + 0x28
7. min_under_threshold_duration	(static)	double-long-unsigned				x + 0x30
8. emergency_profile	(static)	emergency_profile				x + 0x38
9. emergency_profile_group_id_list	(static)	array				x + 0x40
10. emergency_profile_active	(dyn.)	boolean				x + 0x48
11. actions	(static)	action				x + 0x50
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name	Identifie l'instance de l'objet "Limiter" (Limiteur). Voir 6.2.13.

monitored_value	Définit un attribut d'un objet à surveiller. Seuls sont autorisés les attributs ayant des types de donnée simples.
	value_definition::= structure
	{
	class_id: long-unsigned,
	logical_name: octet-string,
	attribute_index: integer
	}

threshold_active	Fournit la valeur de seuil active à laquelle l'attribut surveillé est comparé. threshold: Le "threshold" (seuil) est du même type que

	l'attribut surveillé.
threshold_normal	Fournit la valeur de seuil à laquelle l'attribut surveillé est comparé en fonctionnement normal. threshold: voir ci-dessus.
threshold_emergency	Fournit la valeur de seuil à laquelle l'attribut surveillé est comparé lorsqu'un profil d'urgence est actif. threshold: voir ci-dessus.
min_over_threshold_duration	Définit la durée minimale au-dessus du seuil, en secondes, requise pour exécuter l'action au-dessus du seuil.
min_under_threshold_duration	Définit la durée minimale en dessous du seuil, en secondes, requise pour exécuter l'action en dessous du seuil.

emergency_profile	Un emergency_profile est défini par trois éléments: emergency_profile_id, emergency_activation_time, emergency_duration.
	Un profil d'urgence est activé si l'élément emergency_profile_id concorde avec l'un des éléments de l'emergency_profile _group_id_list, et le temps concorde avec l'emergency_activation_time et l'élément emergency_duration:
	emergency_profile::= structure
	{
	emergency_profile_id: long-unsigned,
	emergency_activation_time: octet-string,
	emergency_duration: double-long-unsigned
	}
	emergency_activation_time définit la date et l'heure auxquelles l'emergency_profile s'est activé. L'octet-string est codé comme spécifié en 4.6.1 pour date_time.
	emergency_duration définit la durée, en secondes, pour laquelle l'emergency_profile est activé.
	Lorsqu'un profil d'urgence est actif, l'attribut emergency_profile_active est mis à TRUE.

emergency_profile_group_id_list	Définit une liste des id de groupe du profil d'urgence. Le profil d'urgence peut être activé uniquement si l'élément emergency_profile_id de l'emergency_profile concorde avec l'un des éléments de l'emergency_profile_group_id-list: array emergency_profile_group_id emergency_profile_group_id: long-unsigned
emergency_profile_active	Indique que l'emergency_profile est actif.

actions	Définit les scripts à exécuter lorsque la valeur surveillée franchit le seuil pendant la durée minimale.
	action::= structure
	{
	action_over_threshold: action_item,
	action_under_threshold: action_item
	}
	où:
	• action_over_threshold définit l'action lorsque la valeur de l'attribut surveillé franchit le seuil dans le sens montant et reste au-dessus du seuil pendant la durée minimale

au-dessus du seuil;

- action_under_threshold définit l'action lorsque la valeur de l'attribut surveillé franchit le seuil dans le sens descendant et reste en dessous du seuil pendant la durée minimale en dessous du seuil;

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

5.4.10 Classe d'interfaces "Sensor manager" (class_id:67, version: 0)

5.4.10.1 Généralités

La plupart des instruments de mesure relevant du domaine d'application de la MID fonctionnent avec des capteurs dédiés (transducteurs et transmetteurs) reliés à l'unité de traitement. Ces capteurs doivent être surveillés en permanence quant à leur fonctionnement et leurs limites pour satisfaire aux exigences métrologiques en vue d'un calcul ultérieur de valeurs monétaires.

De plus, les valeurs mesurées doivent être surveillées. Ces valeurs peuvent être reliées à une grandeur physique – valeurs brutes de tension, de courant, de résistance, de fréquence, de sortie numérique – fournie par le capteur et aux grandeurs mesurées résultant du traitement des informations fournies par le capteur.

Il est nécessaire de surveiller et de journaliser souvent les valeurs pertinentes afin d'obtenir des informations de diagnostic qui permettent:

- l'identification du dispositif capteur;
- la connexion et l'état de scellé du capteur;
- la configuration des capteurs;
- la surveillance du fonctionnement des capteurs;
- la surveillance du résultat du traitement.

La classe d'interfaces "Sensor manager" permet de gérer les informations détaillées relatives à un capteur au moyen d'un seul objet.

Pour des capteurs/dispositifs plus simples, les objets COSEM déjà existants – identifiant les capteurs, détenant les valeurs de mesure et surveillant ces valeurs de mesure – peuvent être utilisés.

5.4.10.2 Spécification de la classe d'interfaces "Sensor manager"

Les instances de l'IC “Sensor manager” gèrent des informations complexes relatives aux capteurs. Elles permettent aussi de surveiller les données brutes et la valeur traitée, dérivée par traitement des données brutes à l'aide d'algorithmes appropriés tels que requis par l'application particulière. Cette IC inclut un certain nombre de fonctions:

- données de plaque signalétique du capteur et informations de site (attributs 2 à 6);
- une fonction “Extended register” pour les données brutes (attributs 7 à 10);
- une fonction “Register monitor” pour les données brutes (attributs 11 à 12);

NOTE 1 Toutes les données brutes (par exemple la tension de sortie d'un capteur de pression) n’ont pas leur propre code/objet OBIS. C'est la raison pour laquelle les données brutes sont incluses dans la classe "Sensor manager".

- une fonction “Register monitor” pour la valeur traitée (attributs 13 à 15).

NOTE 2 Tous les “modules” ne sont pas nécessairement présents. Les attributs non utilisés peuvent ne pas être mis en œuvre ou ne pas être accessibles.

Sensor manager (Gestionnaire de capteur)		**0...n**	**class_id = 67, version = 0**			
Attribut(s)		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. serial_number	(dyn,)	octet-string				x + 0x08
3. metrological_identification	(dyn.)	octet-string				x + 0x10
4. output_type	(dyn.)	enum				x + 0x18
5. adjustment_method	(dyn.)	octet-string				x + 0x20
6. sealing_method	(dyn.)	enum				x + 0x28
7. raw_value	(dyn.)	CHOICE				x + 0x30
8. scaler_unit	(dyn.)	structure				x + 0x38
9. status	(dyn.)	CHOICE				x + 0x40
10. capture_time	(dyn.)	date_time				x + 0x48
11. raw_value_thresholds	(dyn.)	array				x + 0x50
12. raw_value_actions	(dyn.)	array				x + 0x58
13. processed_value	(dyn.)	processed_value_definition				x + 0x60
14. processed_value_thresholds	(dyn.)	array				x + 0x68
15. processed_value_actions	(dyn.)	array				x + 0x70
Méthodes spécifiques		***m/o***				
1. reset (data)		o				x + 0x80

Description d'attribut

logical_name Identifie l'instance de l'objet “Sensor manager” (Gestionnaire de capteurs).

serial_number Identifie le capteur (->informations de site).

NOTE 3 Pour un capteur simple, le numéro de série peut être détenu par les objets Data avec des codes OBIS appropriés s'il s'agit de la seule information requise. Voir les objets d'usage général liés aux gaz, les objets de configuration.

metrological_identification Décrit les informations du capteur pertinentes du point de vue métrologique, par exemple: identificateur métrologique, date d'étalonnage.

output_type décrit la sortie physique du capteur (->informations de site)

enum
0 = non spécifié,
1 = 4–20 mA,
2 = 0–20 mA,
3 = 0–5 V,
4 = 0–10 V,
5 = Pt100,
6 = Pt500,
7 = Pt1 000,
8 = HART
128 = spécifique au fabricant

	Toutes les autres valeurs sont réservées.
adjustment_method	Décrit la méthode d'ajustement du capteur, par exemple par l'équation à trois points de mesure.
sealing_method	Type de scellés appliqués au capteur. enum 0 = aucun, 1 = mécanique (par exemple: fil métallique ou étiquette de protection; 2 = électronique (par exemple: contact); 3 = logiciel (par exemple: protection par mot de passe);
raw_value	Valeur physique provenant du capteur (par exemple: tension provenant d'un convertisseur de pression en tension). Pour les choix possibles de types de données, voir l'IC “Register”.
scaler_unit	Scalaire et l'unité de la valeur brute (raw_value). Pour la définition de scaler_unit, voir l'IC “Register”.
status	Statut de la dernière valeur brute saisie (pour la définition de status (statut), voir l'IC “Extended register”. Le statut conserve les informations comme: • défaillance de capteur, • capteur activé / non activé. La signification des éléments de statut doit être fournie pour chaque instance de l'objet “Sensor manager" (gestionnaire de capteur).
capture_time	Fournit les informations de date et d'heure spécifiques à l'objet "Sensor manager" montrant le moment où la valeur de l'attribut "raw_value" a été saisie. Pour les choix possibles de types de données, voir la classe d'interfaces “Extended register”.
raw_value_thresholds	Fournit les valeurs de seuil auxquelles l'attribut raw_value est comparé. array threshold threshold: Le "threshold" (seuil) est du même type que les "raw-data" (données brutes).
raw_value_actions	Définit les scripts à exécuter lorsque la “raw_value” franchit le seuil correspondant. L'attribut “raw_value_actions” a exactement le même nombre d'éléments que l'attribut “raw_value_thresholds”. L'ordre de classement des action_items correspond à l'ordre de classement des thresholds (seuils). Pour la spécification des actions, voir l'IC “Register monitor”.
processed_value	Référence l'attribut détenant la valeur traitée des données brutes fournies par le capteur.

	processed_value_definition::= structure
	{
	class_id: long_unsigned,
	logical_name: octet-string,
	attribute_index: integer
	}
	A noter que plus d'un objet "Sensor manager" et plus d'un objet "valeur traitée" peuvent appartenir au même capteur.

processed_value_thresholds	Fournit les valeurs de seuil auxquelles la valeur traitée est comparée. array threshold threshold: Le "threshold" (seuil) est du même type que la valeur traitée.
processed_ value_actions	Définit les scripts à exécuter lorsque la valeur traitée franchit le seuil correspondant. L'attribut “processed_value_actions” a exactement le même nombre d'éléments que l'attribut “processed_value_thresholds”. L'ordre de classement des action_items correspond à l'ordre de classement des thresholds (seuils). Pour la spécification des actions, voir l'IC “Register monitor”.

Description de la méthode	
reset (data)	Cette méthode réinitialise la raw_value à la valeur par défaut. La valeur par défaut est une constante spécifique à l'instance. L'attribut capture_time est mis à l'heure de l'exécution de la réinitialisation. data::= integer(0)

5.4.10.3 Exemple pour capteur de pression absolue

La Figure 14 illustre la définition des seuils supérieur et inférieur applicables.

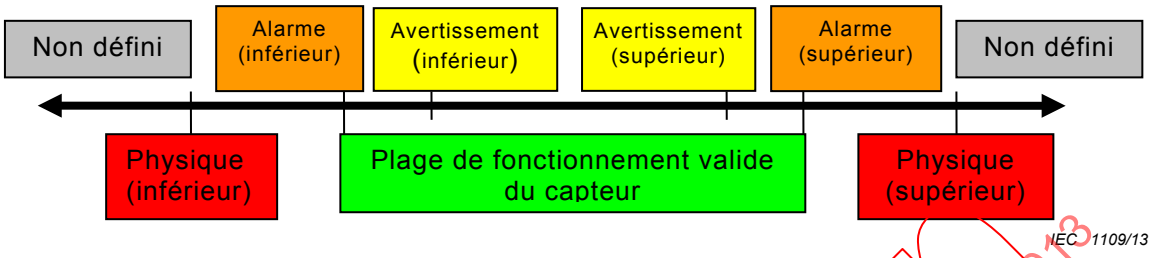


Figure 14 – Définition des seuils supérieur et inférieur

Le Tableau 8 montre un exemple des divers seuils.

Tableau 8 – Présentation explicite des tableaux de valeurs seuils

Threshold (Seuil)	Physique (inférieur)	Physique (supérieur)	Alarme (inférieur)	Alarme (supérieur)	Avertissement (inférieur)	Avertissement (supérieur)
value	1,0	5,5	1,2	5,0	1,4	4,8
scaler_unit	1, Volt	1, Volt	1, bar	1, bar	1, bar	1, bar

Le Tableau 9 montre un exemple d'actions à exécuter lorsque les seuils sont franchis.

Tableau 9 – Présentation explicite d'action_sets

action_set	Physique (inférieur)	Physique (supérieur)	Alarme (inférieur)	Alarme (supérieur)	Avertisse-ment (inférieur)	Avertisse-ment (supérieur)
action_up	clr_phys_alarm_bit	set_phys_alarm_bit	clr_alarm_bit	set_alarm_bit	clr_warn_bit	set_warn_bit
action_down	set_phys_alarm_bit	clr_phys_alarm_bit	set_alarm_bit	clr_alarm_bit	set_warn_bit	clr_warn_bit

5.5 Classes d'interfaces pour l'établissement de l'échange de données via des ports locaux et des modems

5.5.1 IEC local port setup (class_id: 19, version: 1)

Cette IC permet de modéliser la configuration de ports de communication en utilisant les protocoles spécifiés dans la CEI 62056-21. Plusieurs ports peuvent être configurés.

Établissement de port local CEI		0...n	class_id = 19, version = 1			
Attributs		*Type de donnée*	*Min.*	*Max.*	*Def.*	Nom court
1. logical_name	(static)	octet-string				x
2. default_mode	(static)	enum				x + 0x08
3. default_baud	(static)	enum				x + 0x10
4. prop_baud	(static)	enum				x + 0x18
5. response_time	(static)	enum				x + 0x20
6. device_addr	(static)	octet-string				x + 0x28
7. pass_p1	(static)	octet-string				x + 0x30
8. pass_p2	(static)	octet-string				x + 0x38
9. pass_w5	(static)	octet-string				x + 0x40
Méthodes spécifiques		*m/o*				

Description d'attribut

logical_name

Identifie l'instance d'objet “IEC local port setup”. Voir 6.2.14.

default_mode

Définit le protocole utilisé par le compteur sur le port.

enum:
(0) protocole selon la CEI 62056-21 (modes A...E),
(1) protocole selon la CEI 62056-46. En utilisant cette valeur d'énumération, tous les autres attributs de cette IC ne sont pas applicables.
(2) protocole non spécifié. En utilisant cette valeur d'énumération, l'attribut 4), prop_baud, est utilisé pour régler la vitesse de communication sur le port. Tous les autres attributs ne sont pas applicables.

default_baud

Définit le débit en bauds pour la séquence d'ouverture.

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

prop_baud

Définit la vitesse en bauds devant être proposée par le compteur.

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

response_time	Définit le temps minimum entre la réception d'une requête (fin de télégramme de la requête) et l'émission de la réponse (début de télégramme de réponse). enum: (0) 20 ms, (1) 200 ms
device_addr	Adresse de dispositif selon la CEI 62056-21. .
pass_p1	Mot de passe 1 selon la CEI 62056-21. .
pass_p2	Mot de passe 2 selon la CEI 62056-21. .
pass_w5	Mot de passe W5 réservé pour des applications nationales.

5.5.2 IEC HDLC setup (class_id: 23, version: 1)

Cette IC permet de modéliser et configurer des voies de communications selon la CEI 62056-46. Plusieurs voies de communications peuvent être configurées.

IEC HDLC setup (Configuration de HDLC CEI)		**0...n**	**class_id = 23, version = 1**			
Attributs		***Type de donnée***	**Min.**	**Max.**	**Def.**	**Nom court**
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. window_size_transmit	(static)	unsigned	1	7	1	x + 0x10
4. window_size_receive	(static)	unsigned	1	7	1	x + 0x18
5. max_info_field_length_transmit	(static)	long-unsigned	32	2030	128	x + 0x20
6. max_info_field_length_receive	(static)	long-unsigned	32	2030	128	x + 0x28
7. inter_octet_time_out	(static)	long-unsigned	20	6000	25	x + 0x30
8. inactivity_time_out	(static)	long-unsigned	0		120	x + 0x38
9. device_address	(static)	long-unsigned	0x0010	0x3FFD		x + 0x40
Méthodes spécifiques		***m/o***				

NOTE 1 La valeur maximale des attributs max_info_field_length_transmit et max_info_field_length_receive a été augmentée de 128 à 2 030 pour des raisons d'efficacité.

NOTE 2 Un max_info_field_length_receive de 128 octets est nécessaire afin d'assurer une performance minimale.

NOTE 3 La valeur maximale de l'attribut inter-octet-time-out a été augmentée de 1 000 ms à 6 000 ms afin de permettre l'utilisation de supports de communications, lorsqu'il peut se produire de longs retards. La valeur par défaut a été changée à 25 ms pour l'aligner avec 6.4.4.3.3 de la CEI 62056-46:2002CEI 62056-46.

Description d'attribut

Logical_name	Identifie l'instance de l'objet “IEC HDLC setup” (Configuration de HDLC CEI). Voir 6.2.16.
comm_speed	La vitesse de communication prise en charge par le port correspondant. enum: (0) 300 bauds, (1) 600 bauds, (2) 1 200 bauds, (3) 2 400 bauds, (4) 4 800 bauds, (5) 9 600 bauds, (6) 19 200 bauds, (7) 38 400 bauds, (8) 57 600 bauds, (9) 115 200 bauds Cette vitesse de communication peut être neutralisée si le passage au mode HDLC d'un dispositif se fait par le biais d'un mode spécial d'un autre protocole.
window_size_ transmit	Le nombre maximal de trames qu'un dispositif ou système peut émettre avant qu'il ait besoin de recevoir un acquittement provenant de la station correspondante. D'autres valeurs peuvent être négociées pendant l'ouverture de session.
window_size_ receive	Le nombre maximal de trames qu'un dispositif ou système peut recevoir avant qu'il ait besoin d'émettre un acquittement vers la station correspondante. D'autres valeurs peuvent être négociées pendant l'ouverture de session.
max_info_length_ transmit	La longueur maximale du champ information qu'un dispositif peut émettre. Une valeur inférieure peut être négociée pendant l'ouverture de session.
max_info_length_ receive	La longueur maximale du champ information qu'un dispositif peut recevoir. Une valeur inférieure peut être négociée pendant l'ouverture de session.
inter_octet_time_ out	Définit le temps, en millisecondes, au-delà duquel, si aucun autre caractère n’est reçu en provenance de la station primaire, le dispositif traitera les données déjà reçues comme une trame complète.
inactivity_time_out	Définit le temps, en secondes, au-delà duquel, si aucune trame n’est reçue en provenance de la station primaire, le dispositif traitera une déconnexion. Lorsque cette valeur est mise à 0, cela signifie que l'inactivity_time_out n'est pas opérationnel.

device_address	Contient l'adresse de dispositif physique d'un dispositif. Dans le cas de l'adressage d'un octet: 0x00 Adresse NO_STATION, 0x01…0x0F Réservé pour usage futur, 0x10…0x7D Espace d'adresses utilisable, 0x7E Adresse de dispositif 'CALLING', 0x7F Adresse de diffusion Dans le cas de l'adressage de deux octets: 0x0000 Adresse NO_STATION, 0x0001..0x000F Réservé pour usage futur, 0x0010..0x3FFD Espace d'adresses utilisable, 0x3FFE Adresse de dispositif physique 'CALLING', 0x3FFF Adresse de diffusion

5.5.3 IEC twisted pair (1) setup (class_id: 24, version: 0)

Cette IC permet de modéliser et configurer des voies de communications selon le Projet CEI 62056-3-1[6]. Plusieurs voies de communications peuvent être configurées.

IEC twisted pair (1) setup (établissement de paire torsadée (1) CEI)		**0…n**	**class_id = 24, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. secondary_address	(static)	octet-string				x + 0x08
3. primary_address_list	(static)	primary_address_list_type				x + 0x10
4. tabi_list	(static)	tabi_list_type				x + 0x18
5. fatal_error	(dyn.)	enum				x + 0x20
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name	Identifie l'instance d'objet "IEC twisted pair setup". Voir 6.2.17.
secondary_address	Mémorise l'ADS de la station secondaire (voir Projet CEI 62056-3-1) qui correspond à l'équipement réel. octet-string (SIZE(6))

primary_address_list	Mémorise la liste des ADP ou adresses physiques de station primaire pour lesquelles chaque dispositif logique de l'équipement réel (la station secondaire) a été programmé (voir Projet CEI 62056-3-1).
	
	primary_address_list_type::= array
	{
	primary_address_element
	}
	primary_address_element::= octet-string (SIZE(1))

tabi_list	Représente la liste des TAB(i) pour lesquels l'équipement réel (la station secondaire) a été programmé dans le cas d'un appel de station oublié (voir Projet CEI 62056-3-1).

[6] À publier.

tabi_list_type::= array tabi_element

tabi_element::= integer

fatal_error

Représente la dernière occurrence de l'une des erreurs fatales des protocoles décrits dans le Projet CEI 62056-3-1.

La valeur par défaut initiale de cette variable est "00"H. Ensuite, chaque erreur fatale est marquée.

enum:
(0) No-error (absence d'erreur),
(1) t-EP-1F,
(2) t-EP-2F,
(3) t-EL-4F,
(4) t-EL-5F,
(5) eT-1F,
(6) eT-2F,
(7) e-EP-3F,
(8) e-EP-4F,
(9) e-EP-5F,
(10) e-EL-2F

5.5.4 Modem configuration (class_id: 27, version: 1)

Cette IC permet de modéliser la configuration et l'initialisation des modems utilisés pour le transfert de données en provenance/à destination d'un dispositif. Plusieurs modems peuvent être configurés.

Modem configuration (configuration de modem)		**0..n**	**class_id = 27, version = 1**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. comm._speed	(static)	enum	0	9	5	x + 0x08
3. initialization_string	(static)	array				x + 0x10
4. modem_profile	(static)	array				x + 0x18
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name

Identifie l'instance de l'objet “Modem configuration” (Configuration de modem). Voir 6.2.5.

comm_speed

La vitesse de communication entre le dispositif et le modem, pas nécessairement la vitesse de communication sur le réseau étendu.

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

initialization_string	Contient toutes les commandes d'initialisation nécessaires devant être envoyées au modem afin de le configurer correctement. Cela peut inclure la configuration de caractéristiques spéciales du modem.
	
	array initialization_string_element
	
	initialization_string_element::= structure
	{
	request: octet-string,
	response: octet-string,
	delay_after_response: long-unsigned
	}
	Si la matrice contient plus d'un initialization_string_element, les demandes peuvent être envoyées en séquence. La demande suivante est envoyée après la réponse attendue correspondant à la demande précédente et après une attente d'un temps delay_after_response [ms], pour permettre au modem d'exécuter la demande.
	
	NOTE Il est supposé que le modem est préconfiguré et accepte, de ce fait, l'initialization_string (c'est-à-dire chaîne d'initialisation). Si l'initialisation n'est pas nécessaire, la chaîne d'initialisation est vide.

modem_profile	Définit la mise en correspondance des commandes/réponses de la norme Hayes avec les chaînes spécifiques au modem. array modem_profile_element modem_profile_element::= octet-string Le tableau modem_profile doit contenir les chaînes correspondantes pour le modem utilisé dans l'ordre suivant: Élément 0: OK, Élément 1: CONNECT, Élément 2: RING, Élément 3 NO CARRIER, Élément 4: ERROR, Élément 5: CONNECT 1 200, Élément 6 NO DIAL TONE, Élément 7: BUSY, Élément 8: NO ANSWER, Élément 9: CONNECT 600, Élément 10: CONNECT 2 400, Élément 11: CONNECT 4 800, Élément 12 CONNECT 9 600, Élément 13: CONNECT 14 400, Élément 14: CONNECT 28 800, Élément 15: CONNECT 33 600, Élément 16: CONNECT 56 000

5.5.5 Auto answer (class_id: 28, version: 0)

Cette IC permet de modéliser la manière dont le dispositif gère la fonction “Auto answer” du modem, à savoir la réponse aux appels entrants. Plusieurs modems peuvent être configurés.

Auto answer (réponse automatique)		**0...n**	**class_id = 28, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. listening_window	(static)	array				x + 0x10
4. status	(dyn.)	enum				x + 0x18
5. number_of_calls	(static)	unsigned				x + 0x20
6. number_of_rings	(static)	nr_rings_type				x + 0x28
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name

Identifie l'instance de l'objet “Auto answer” (Réponse automatique). Voir 6.2.5.

mode

Définit le mode de fonctionnement de la ligne lorsque le dispositif est en mode réponse automatique.

enum: (0) ligne dédiée au dispositif,

(1) gestion de ligne partagée avec un nombre limité d'appels autorisés. Une fois que le nombre d'appels est atteint, le statut de la fenêtre devient "inactif" jusqu'à la prochaine date de début, quel que soit le résultat de l'appel,

(2) gestion de ligne partagée avec un nombre limité d'appels réussis autorisés. Une fois que le nombre de communications réussies est atteint, l'état de la fenêtre devient "inactif" jusqu'à la prochaine date de début,

(3) actuellement aucun modem connecté,

(200...255) modes spécifiques au fabricant.

listening_window

Contient l'instant de début et de fin où la fenêtre devient active (pour l'instant de début) et inactive (pour l'instant de fin). Le start_date définit implicitement la période.

EXEMPLE Lorsque le jour du mois n'est pas spécifié (égal à 0xFF), cela signifie qu'il s'agit de la gestion d'une ligne de partage quotidien. Une gestion de fenêtre quotidienne, mensuelle ...peut être définie.

array window_element

window_element::= structure

{

start _time: octet-string,

end_time: octet-string

}

start_time et end_time sont formatés comme établi en 4.6.1 pour *date_time*

status

Ici est défini l'état de la fenêtre.

	enum: (0)	Inactive: le dispositif ne gérera aucun nouvel appel entrant. Cet état est automatiquement réinitialisé à "Active" lorsque la fenêtre d'écoute suivante commence,
	(1)	Active: le dispositif peut répondre au nouvel appel entrant,
	(2)	Locked (verrouillée): Cette valeur peut être fixée automatiquement par le dispositif ou par un client spécifique lorsque ce client a terminé sa session de lecture et souhaite rendre la ligne au client avant la fin de la durée de la fenêtre. Cet état est automatiquement réinitialisé à "Active" lorsque la fenêtre d'écoute suivante commence.
number_of_calls	Ce nombre est la référence utilisée dans les modes 1 et 2. Lorsqu'il est mis à 0, cela signifie qu'il n'y a pas de limite.	

number_of_rings	Définit le nombre de sonneries avant que le compteur ne connecte le modem. Deux cas sont distingués: nombre de sonneries dans la fenêtre définie par l'attribut “listening_window” et nombre de sonneries à l'extérieur de la “listening_window”.
	
	nr_rings_type::= structure
	{
	nr_rings_in_window: unsigned, (0: pas de connexiondans la fenêtre)
	nr_rings_out_of_window: unsigned (0: pas de connexion hors de la fenêtre)
	}	

5.5.6 Auto connect (class_id: 29, version: 1)

Cette IC permet de modéliser la gestion du transfert de données du dispositif vers une ou plusieurs destinations.

Les messages à envoyer, les conditions dans lesquelles ils doivent être envoyés et la relation entre les divers modes, les fenêtres d'appel et les destinations ne sont pas définis ici.

En fonction du mode, une ou plusieurs instances de cette IC peuvent être nécessaires pour accomplir la fonction d'envoi de messages.

Auto connect (connexion automatique)		**0...n**	**class_id = 29, version = 1**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. repetitions	(static)	unsigned				x + 0x10
4. repetition_delay	(static)	long-unsigned				x + 0x18
5. calling_window	(static)	array				x + 0x20
6. destination_list	(static)	array				x + 0x28
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name	Identifie l'instance de l'objet “Auto connect” (Connexion automatique). Voir 6.2.5.
mode	Définit le mode commandant la fonctionnalité de composition automatique de numéros concernant le choix du moment, le type de message à envoyer et l'infrastructure à utiliser. enum: (0) pas de numérotation automatique, (1) numérotation automatique autorisée à tout moment, (2) numérotation automatique autorisée dans la limite du temps de validité de la fenêtre d'appel, (3) numérotation automatique "régulière" autorisée dans la limite du temps de validité de la fenêtre d'appel; numérotation automatique déclenchée par une “alarme” autorisée à tout moment, (4) envoi de SMS via les réseaux mobiles terrestres publics (PLMN), (5) envoi de SMS via le RPTC, (6) envoi de courriel (email), (200..255) modes spécifiques au fabricant
repetitions	Le nombre maximal d'essais dans le cas de tentatives d'appel non réussies.
repetition_delay	Délai en secondes à observer avant qu'une tentative infructueuse puisse être répétée. repetition_delay 0 signifie que le retard n'est pas spécifié.

calling_window	Contient le marqueur de date/heure de début et de fin où la fenêtre devient active (pour l'instant de début) ou inactive (pour l'instant de fin). Le start_date définit implicitement la période.
	
	EXEMPLE Lorsque le jour du mois n'est pas spécifié (égal à 0xFF), cela signifie qu'il s'agit de la gestion d'une ligne de partage quotidien. Une gestion de fenêtre quotidienne, mensuelle ...peut être définie.
	
	array window_element
	window_element::= structure
	{
	start_time: octet-string,
	end_time: octet-string
	}
	start_time et end_time sont formatés comme établi en 4.6.1 pour *date_time*

destination_list	Contient la liste de destinations (par exemple numéros de téléphone, adresses email ou leurs combinaisons) vers lesquelles le(s) message(s) est/sont à envoyer dans certaines conditions. Les conditions et leur lien aux éléments de la matrice ne sont pas définis ici. array destination destination::= octet-string

5.6 Classes d'interfaces pour l'établissement d'échange de données via le M-Bus

5.6.1 M-Bus slave port setup (class_id: 25, version: 0)

NOTE 1 Le nom de cette IC a été changé de "M-BUS port setup" en "M-Bus slave port setup" afin d'indiquer qu'elle sert à établir l'échange de données lorsqu'un serveur COSEM communique avec un client COSEM en utilisant le M-Bus câblé.

Cette IC permet de modéliser et configurer des voies de communications selon l'EN 13757-2. Plusieurs voies de communications peuvent être configurées.

M-Bus slave port setup (établissement de port de M-Bus esclave)	**0...n**	**class_id = 25, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. default_baud (static)	enum	0	5	0	x + 0x08
3. avail_baud (static)	enum	0	7		x + 0x10
4. addr_state (static)	enum				x + 0x18
5. bus_address (static)	unsigned				x + 0x20
Méthodes spécifiques	***m/o***				

Description d'attribut

logical_name

Identifie l'instance d'objet "M-Bus slave port setup" (établissement de port de M-Bus esclave). Voir 6.2.18

default_baud

Définit le débit en bauds pour la séquence d'ouverture.

enum:
(0) 300 bauds,
(3) 2 400 bauds,
(5) 9 600 bauds

avail_baud

Définit les débits en bauds qui peuvent être négociés après démarrage.

enum:
(0) 300 bauds,
(1) 600 bauds,
(2) 1 200 bauds,
(3) 2 400 bauds,
(4) 4 800 bauds,
(5) 9 600 bauds,
(6) 19 200 bauds,
(7) 38 400 bauds

addr_state

Définit si, oui ou non, le dispositif a eu une adresse assignée depuis la dernière mise sous tension du dispositif.

enum:
(0) N'a pas encore reçu d'adresse attribuée,
(1) a une adresse attribuée, soit par réglage manuel, soit par une méthode automatisée.

bus_address

L'adresse actuellement attribuée sur le bus pour le dispositif.

NOTE 2 Si aucune adresse de bus n'est attribuée, la valeur est 0.

5.6.2 M-Bus client (class_id: 72, version: 0)

Les instances de cette IC permettent d'installer des dispositifs de M-Bus esclaves et d'échanger des données avec eux. Chaque objet "M-Bus client" commande un dispositif M-Bus esclave. Pour les détails relatifs à la couche d'application M-Bus dédiée voir EN 13757-3.

Le dispositif client M-Bus peut avoir une ou plusieurs interfaces physiques M-Bus, qui peuvent être configurées à l'aide d'instances de l'IC "M-Bus master port setup", voir 5.6.4.

Un dispositif de M-Bus esclave est identifié avec son adresse primaire Primary Address, son numéro d'identification Identification Number, son identificateur de fabricant Manufacturer ID etc. comme défini dans l'EN 13757-3:2004, Article 5, *Variable Data respond* (Réponse de données variables). Ces paramètres sont contenus dans les attributs respectifs de l'IC M-Bus client, voir 5.6.2.

Les valeurs à saisir à partir d'un dispositif M-Bus esclave sont identifiées par l'attribut capture_definition, contenant une liste d'identificateurs de données (DIB, VIB) pour le dispositif M-Bus.esclave. Les données sont saisies périodiquement ou sur un déclencheur approprié. Chaque élément de donnée est stocké dans un objet "M-Bus value" (valeur M-Bus) de l'IC “Extended register”. Les objets "M-Bus value" peuvent être saisis dans les objets génériques "M-Bus Profile", éventuellement avec d'autres objets, non spécifiques à M-Bus.

En utilisant les méthodes d'objets "M-Bus client", les dispositifs M-Bus esclaves peuvent être installés et désinstallés.

Il est également possible d'envoyer des données à des dispositifs M-Bus esclaves et d'exécuter des opérations telles que la réinitialisation d'alarmes, le réglage d'horloge, le transfert d'une clé de chiffrement, etc.

M-Bus client (M-Bus Client)		**0...n**	**class_id = 72, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	***Nom court***
1. logical_name	(static)	octet-string				x
2. mbus_port_reference	(static)	octet-string				x + 0x08
3. capture_definition	(static)	array				x + 0x10
4. capture_period	(static)	double-long-unsigned				x + 0x18
5. primary_address	(dyn.)	unsigned				x + 0x20
6. identification_number	(dyn.)	double-long-unsigned				x + 0x28
7. manufacturer_id	(dyn.)	long-unsigned				x + 0x30
8. version	(dyn.)	unsigned				x + 0x38
9. device_type	(dyn.)	unsigned				x + 0x40
10. access_number	(dyn.)	unsigned				x + 0x48
11. status	(dyn.)	unsigned				x + 0x50
12. alarm	(dyn.)	unsigned				x + 0x58
Méthodes spécifiques		***m/o***				
1. slave_install (data)		o				x + 0x60
2. slave_deinstall (data)		o				x + 0x68
3. capture (data)		o				x + 0x70
4. reset_alarm (data)		o				x + 0x78
5. synchronize_clock (data)		o				x + 0x80
6. data_send (data)		o				x + 0x88
7. set_encryption_key (data)		o				x + 0x90
8. transfer_key (data)		o				x + 0x98

Description d'attribut

logical_name	Identifie l'instance de l'objet “M-Bus client” (M-Bus client). Pour les noms logiques, voir 6.2.18.
mbus_port_reference	Fournit une référence à un objet “M-Bus master port setup”, utilisé pour configurer un port M-Bus, chaque interface permettant d'échanger des données avec un ou plusieurs dispositifs M-Bus esclaves.

capture_definition	Fournit la capture_definition pour les dispositifs M-Bus esclaves.
	
	array capture_definition_element
	
	capture_definition_element::= structure
	
	{
	data_information_block: octet-string,
	value_information_block: octet-string
	}
	NOTE 1 Les éléments data_information_block et value_information_block correspondent respectivement à Data Information Block (DIB) et à Value Information Block (VIB) décrits dans l'EN 13757-3:2004, 6.2 et Article 7.

capture_period	>= 1: Saisie automatique supposée. Spécifie la période de saisie en secondes. 0: Absence de saisie automatique: la saisie est déclenchée extérieurement ou des événements de saisie se produisent de manière asynchrone.
primary_address	Contient l'adresse primaire du dispositif M-Bus esclave, dans la plage 0...250. Si le dispositif esclave est déjà configuré et donc son adresse primaire est différente de 0, cela doit être écrit dans l'attribut primary_address. À partir de cet instant, l'échange de données avec le dispositif de M-Bus esclave est possible. Autrement, la méthode slave_install doit être utilisée; voir ci-dessous. NOTE 2 L'attribut primary_address ne peut pas être utilisé pour stocker une adresse primaire souhaitée pour un dispositif esclave non configuré. Si l'attribut adresse primaire est positionné, cela signifie que le client de M-Bus peut opérer immédiatement avec cette adresse primaire, ce qui n'est pas le cas avec un dispositif esclave non configuré.
identification_number	Contient l'élément "Identification Number" (numéro d'identification) de l'en-tête de donnée tel que spécifié dans l'EN 13757-3:2004, 5.4. Il s'agit soit d'un numéro de fabrication fixe, soit d'un numéro modifiable par le client, codé avec 8 chiffres BCD condensés (4 octets) et qui se situe donc entre 00 000 000 et 99 999 999. Il peut être préréglé au moment de la fabrication avec un numéro unique mais pourrait être modifiable plus tard, notamment s'il est fourni en plus un numéro de fabrication unique et non modifiable (DIF = 0Ch, VIF = 78h), voir 7.2.
manufacturer_id	Contient l'élément "Manufacturer Identification" (identification du fabricant) de l'en-tête de donnée tel que spécifié dans

	l'EN 13757-3:2004, 5.5. Il s'agit d'un mot binaire non signé de 2 octets. Ce manufacturer_id est calculé à partir du code ASCII code de l'ID de fabricant selon la CEI 62056-21 (trois lettres majuscules), en utilisant la formule spécifiée dans l'EN 13757-3:2004, 5.5.
version	Contient l'élément "Version" de l'en-tête de donnée tel que spécifié dans l'EN 13757-3:2004, 5.6. Il spécifie la génération ou la version du compteur et dépend du fabricant. Il peut servir à s'assurer que le numéro d'identification est unique dans chaque numéro de version.
device_type	Contient l'élément "Device type identification " (identification du type de dispositif) de l'en-tête de donnée tel que spécifié dans l'EN 13757-3:2004, 5.7, Tableau 3.
access_number	Contient l'élément "Access Number " (numéro d'accès) de l'en-tête de donnée tel que spécifié dans l'EN 13757-3:2004, 5.8. Il a un mot binaire non signé et il est incrémenté (modulo 256) de 1 avant ou après chaque RSP-UD provenant de l'esclave. Sachant qu'il peut aussi servir à permettre à des utilisateurs finals privés de détecter une indésirable lecture excessivement fréquente de leurs compteurs de consommation, il convient qu'il ne soit pas réinitialisable par une quelconque communication de bus.
status	Contient l'élément "Status byte" (octet d'état) de l'en-tête de donnée tel que spécifié dans l'EN 13757-3:2004, 5.9, Tableaux 4 et 5.
alarm	Contient l'état Alarm spécifié dans l'EN 13757-3:2004, Annexe D. Il est codé avec le type de donnée D (Boolean, dans ce cas 8 bits). Les bits mis signalent les bits d'alarme ou les codes d'alarme. La signification de ces bits est spécifique au fabricant.
Description de la méthode	
slave_install (data)	Installe un dispositif esclave qui est encore non configuré (son adresse primaire est 0). Cette méthode ne peut être invoquée avec succès que si la valeur de l'attribut primary_address est 0. Les actions suivantes sont entreprises: • L'adresse M-Bus 0 est vérifiée pour détecter la présence d'un nouveau dispositif. • Si aucun M-Bus esclave non installé n'est trouvé, l'invocation de la méthode échoue; • Si la méthode slave_install est invoquée sans paramètre, l'adresse primaire est affectée automatiquement. Cela se fait en vérifiant l'attribut primary_address de tous les objets "M-Bus client" dans le dispositif DLMS/COSEM et en sélectionnant ensuite le premier numéro non utilisé. L'attribut primary_address est mis à cette adresse et il est ensuite transféré au dispositif M-Bus esclave; • Si la méthode slave_install est invoquée avec une adresse primaire comme paramètre, l'attribut primary_address est mis à cette valeur et il est ensuite transféré au dispositif M-Bus esclave.

	data::= unsigned (aucune donnée, ou une adresse primaire valide) NOTE 3 Les dispositifs esclaves non configurés sont configurés avec l'adresse primaire telle que spécifiée dans l'EN 13757-3:2004, E.5.
slave_deinstall (data)	Désinstalle le dispositif esclave. Le but principal de ce service est de désinstaller le dispositif de M-Bus esclave et de préparer le maître pour l'installation d'un nouveau dispositif. Les actions suivantes sont entreprises: • l'adresse M-Bus est mise à 0 dans le dispositif M-Bus esclave; • la clé de chiffrement transférée précédemment au dispositif M-Bus esclave est détruite; la clé par défaut n'est pas altérée; • l'attribut "encryption status" (état de chiffrement) doit être réinitialisé à 0; • l'attribut primary_address est également mis à 0. NOTE 4 Un nouvel esclave M-Bus peut être installé seulement une fois que la valeur de l'attribut primary_address est 0. data::= integer(0)
capture (saisie)	Capture des valeurs – telles que spécifiées par l'attribut capture_definition – provenant du dispositif M-Bus esclave. data::= integer(0)
reset_alarm	Réinitialise l'état d'alarme du dispositif M-Bus esclave. data::= integer(0)
synchronize_clock	Synchronise l'horloge du dispositif M-Bus esclave avec celle du dispositif de M-Bus client. data::= integer(0)
data_send	Envoie des données au dispositif M-Bus esclave. data array data_definition_element data_definition_element::= structure { data_information_block: octet-string, value_information_block : octet-string data: CHOICE { -- types de données simples null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], UTF8-string [12], bcd [13],

	integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] } }
set_encryption_key	Met en place la clé de chiffrement à utiliser avec le dispositif M-Bus esclave. La modification du chiffrement requiert deux étapes: En premier lieu, la clé est envoyée à l'esclave M-Bus, chiffrée avec la clé principale, en utilisant la méthode transfer_key. En second lieu, la clé est installée dans le maître M-Bus en utilisant la méthode set_encryption_key. data::= octet-string (encryption_key) Après l'installation de l'esclave M-Bus, le client de M-Bus détient une clé de chiffrement vide. Ainsi, le chiffrement des télégrammes de M-Bus est désactivé. Le chiffrement peut être désactivé en invoquant la méthode set_encryption_key avec des données null comme paramètre.
transfer_key	Transfère une clé de chiffrement à l'esclave M-Bus. data::= octet-string (encrypted_key) Chaque dispositif de M-Bus esclave doit être délivré avec une clé de chiffrement par défaut. Avant que des télégrammes de M-Bus chiffrés ne puissent être utilisés, une clé de chiffrement opérationnelle doit être envoyée à l'esclave M-Bus, par invocation de la méthode transfer_key. Le paramètre d'invocation de la méthode est la clé de chiffrement opérationnelle chiffrée avec la clé par défaut du dispositif de M-Bus esclave. Le télégramme de M-Bus envoyé n'est pas chiffré. Après l'exécution, le chiffrement est activé et tous les télégrammes ultérieurs sont chiffrés. Une nouvelle clé de chiffrement peut être installée dans le client en invoquant la méthode set_encryption_key avec la nouvelle clé de chiffrement comme paramètre. Avec des invocations ultérieures de la méthode transfer_key, de nouvelles clés de chiffrement peuvent être envoyées à l'esclave de M-Bus. Le paramètre d'invocation de la méthode est la nouvelle clé de chiffrement chiffrée avec la clé par défaut. Le télégramme de M-Bus est chiffré. Lorsqu'un M-Bus esclave est désinstallé, la clé de chiffrement est détruite, mais la clé par défaut n'est pas altérée. Le chiffrement reste désactivé jusqu'à ce qu'un nouveau chiffrement soit transféré.

5.6.3 Wireless Mode Q channel (class_id: 73, version: 1)

Les instances de cette IC définissent les paramètres opérationnels pour la communication en utilisant les interfaces de mode Q. Voir également l'EN 13757-3.

Wireless Mode Q channel (voie en mode Q sans fil)	**0...n**	**class_id = 73, version = 1**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. addr_state (static)	enum				x + 0x08
3. device_address (static)	octet-string				x + 0x10
4. address_mask (static)	octet-string				x + 0x18
Méthodes spécifiques	***m/o***				

Description d'attribut

addr_state	Définit si, oui ou non, le dispositif a eu une adresse attribuée depuis la dernière mise sous tension du dispositif enum: (0) n'a pas encore reçu d'adresse attribuée, (1) a une adresse attribuée, soit par réglage manuel, soit par une méthode automatisée
device_address	L'adresse actuellement attribuée du dispositif sur le réseau
address_mask	L'adresse de groupe à laquelle le dispositif répondra lorsque l'adressage de forme courte est utilisé

5.6.4 M-Bus master port setup (class_id: 74, version: 0)

Les instances de cette IC définissent les paramètres opérationnels pour la communication en utilisant les interfaces de l'EN 13757-2 si le dispositif agit comme un M-Bus maître.

M-Bus master port setup (établissement de port de M-Bus maître)	**0...n**	**class_id = 74, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. comm_speed (static)	enum	0	7	3	x + 0x08
Méthodes spécifiques	***m/o***				

Description d'attribut

logical_name	Identifie l'instance d'objet "M-Bus master port setup" (établissement de port M-Bus maître). Voir 6.2.18.
comm_speed	La vitesse de communication prise en charge par le port enum: (0) 300 bauds, (1) 600 bauds, (2) 1 200 bauds, (3) 2 400 bauds, (4) 4 800 bauds, (5) 9 600 bauds, (6) 19 200 bauds,

(7) 38 400 bauds

5.7 Classes d'interfaces pour l'établissement d'échange de données sur Internet

5.7.1 TCP-UDP setup (class_id: 41, version: 0)

Cette IC permet de modéliser l'établissement de la sous-couche TCP ou UDP de la couche transport COSEM basée sur TCP ou UDP d'un profil de communication basé sur TCP-UDP/IP.

Dans les profils de communication basés sur TCP-UDP/IP, toutes les AA entre un dispositif physique hébergeant un ou plusieurs processus d'application client COSEM et un dispositif physique hébergeant un ou plusieurs AP serveur COSEM s'appuient sur une seule connexion TCP ou UDP. L'entité TCP ou UDP est enveloppée dans la couche transport COSEM basée sur TCP-UDP. Au sein d'un dispositif physique, chaque Processus d'Application – AP client ou dispositif logique serveur – est lié à un Wrapper Port (WPort), port conteneur. La liaison se fait avec l'aide de l'objet "SAP Assignment". Voir 5.3.3.

D'autre part, une couche transport COSEM basée sur TCP ou sur UDP peut être capable de prendre en charge plus d'une connexion TCP ou UDP, entre un dispositif physique et plusieurs dispositifs physiques homologues hébergeant des AP COSEM.

Lorsqu'un dispositif physique COSEM prend en charge diverses couches liaisons de données – par exemple Ethernet et PPP – une instance de l'objet "TCP-UDP setup" est nécessaire pour chacune d'elles.

TCP-UDP setup (établissement de TCP-UDP)		**0...n**	**class_id = 41, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. TCP-UDP_port	(static)	long-unsigned				x + 0x08
3. IP_reference	(static)	octet-string				x + 0x10
4. MSS	(static)	long-unsigned	40	65 535	576	x + 0x18
5. nb_of_sim_conn	(static)	unsigned	1			x + 0x20
6. inactivity_time_out	(static)	long-unsigned			180	x + 0x28
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name Identifie l'instance de l'objet “TCP-UDP setup” (Établissement de TCP-UDP). Voir 6.2.19.

TCP-UDP_port Détient le numéro de port TCP-UDP sur lequel le dispositif physique est à l'écoute pour détecter l'application DLMS/COSEM.

Pour DLMS/COSEM, les numéros de port suivants ont été enregistrés par l'IANA. Voir http://www.iana.org/assignments/port-numbers (dernière mise à jour le 2010-08-17)

- dlms/cosem 4059/TCP DLMS/COSEM
- dlms/cosem 4059/UDP DLMS/COSEM

IP_reference Référence un objet "IP setup" (établissement d'IP) par son nom logique. L'objet référencé contient des informations relatives aux

	réglages de l'adresse IP de la couche IP prenant en charge la couche TCP-UDP.
MSS	Avec l'aide de l'option Taille de segment maximale (Maximum Segment Size, MSS), une couche TCP peut indiquer la taille maximale de segment de réception à son partenaire. A noter que: • cette option doit seulement être envoyée dans la demande de connexion initiale (à savoir en des segments avec le bit de commande SYN envoyé); • Si cette option est absente, la MSS est conventionnellement considérée comme sa valeur par défaut, 576; • La MSS n'est pas négociable; sa valeur est indiquée par son attribut.
nb_of_sim_conn	Le nombre maximal de connexions simultanées que la couche transport COSEM basée sur TCP-UDP est capable de prendre en charge.
inactivity_time_out	Définit le temps, en secondes, sur lequel, si aucune trame n'est reçue en provenance du client COSEM, la connexion TCP inactive doit être abandonnée. Lorsque cette valeur est mise à 0, cela signifie que l'inactivity_time_out n'est pas opérationnel. En d'autres termes, une connexion TCP, une fois établie, dans des conditions normales – pas de coupure secteur, etc. – ne sera jamais abandonnée par le serveur COSEM. A noter que toutes les actions liées à la gestion de la fonction de dépassement de délai d'inactivité – mesure du temps d'inactivité, abandon de la connexion TCP si la temporisation est terminée, etc. – sont gérées dans la mise en œuvre de la couche TCP-UDP.

5.7.2 IPv4 setup (class_id: 42, version: 0)

Cette IC permet de modéliser l'établissement de la couche IPv4, de gérer toutes les informations liées aux réglages d'adresses IP associés à un dispositif donné et à une connexion de couche inférieure sur laquelle ces réglages sont utilisés.

Il doit y avoir une instance de cette IC dans un dispositif pour chaque interface réseau différente mise en œuvre. Par exemple, si un dispositif a deux interfaces (utilisant le profil TCP/IP sur les deux), il doit y avoir deux instances de l'IC "IPv4 setup" dans le dispositif en question: une pour chacune de ces interfaces.

IPv4 setup (établissement de IPv4)	**0...n**	**class_id = 42, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. DL_reference (static)	octet-string				x + 0x08
3. IP_address	double-long-unsigned				x + 0x10
4. multicast_IP_address	array				x + 0x18
5. IP_options	array				x + 0x20
6. subnet_mask	double-long-unsigned				x + 0x28
7. gateway_IP_address	double-long-unsigned				x + 0x30
8. use_DHCP_flag (static)	boolean				x + 0x38
9. primary_DNS_address	double-long-unsigned				x + 0x40
10. secondary_DNS_address	double-long-unsigned				x + 0x48
Méthodes spécifiques	***m/o***				
1. add_mc_IP_address (data)	o				x + 0x60
2. delete_mc_IP_address (data)	o				x + 0x68
3. get_nbof_mc_IP_addresses (data)	o				x + 0x70

Description d'attribut

logical_name

Identifie l'instance de l'objet "IPv4 setup" (Établissement de IPv4). Voir 6.2.19.

DL_reference

Référence un objet d'établissement de couche liaison de données (par exemple Ethernet ou PPP) par son nom logique. L'objet référencé contient des informations relatives aux réglages spécifiques de la couche liaison de données prenant en charge la couche IP.

IP_address

Contient la valeur de l'adresse IP (IPv4) de ce dispositif physique sur le réseau auquel le dispositif est connecté via l'interface de couche inférieure référencée.

Il peut être (statique) ou (dynamique). Dans le second cas, l'affectation d'adresse IP dynamique (par exemple DHCP) est utilisée.

Si aucune adresse IP n'est assignée, la valeur est 0.

multicast_IP_address

Contient un tableau d'adresses IP. Les adresses IP dans ce tableau doivent s'inscrire dans la plage d'adresses de groupe de multidiffusion (adresses de "Class D", y compris les adresses IP dans la plage de 224.0.0.0 à 239.255.255.255).

Lorsqu'un dispositif reçoit un datagramme IP avec l'une de ces adresses IP dans le champ adresse IP de destination, il doit considérer que ce datagramme lui est adressé.

multicast_IP_address::= array double-long-unsigned

IP_options

Contient les paramètres nécessaires à la prise en charge des options IP sélectionnées – par exemple horodatage de datagrammes ou

services de sécurité (IPSec).

IP_options::= array IP_options_element

IP_options_element::= SEQUENCE

{

IP-Option-Type: unsigned,
IP-Option-Length: unsigned,
IP-Option-Data: octet-string

}

IP-Option-Types autorisés:

- Security - 0x82

Si cette option est présente, le dispositif doit être autorisé à envoyer des paramètres de sécurité, de compartimentage, de restrictions de gestion et de TCC (groupe d'utilisateurs fermé) dans ses datagrammes IP. La valeur du champ IP-Option-Length doit être 11, le IP-Option-Data doit contenir la valeur des données de sécurité cloisonnement, gestion des restrictions et le code de pays de la transmission tels que spécifiés dans STD 0005 / RFC 791.

– Loose Source et Record Route - 0x83

Si cette option est présente, le dispositif doit fournir des informations de routage devant être utilisées par les passerelles pour transmettre le datagramme à la destination et pour enregistrer les informations de chemin.

Les valeurs de IP-Option-length et IP-Option-Data sont spécifiées dans STD 0005 / RFC 791.

– Strict Source et Record Route - 0x89

Si cette option est présente, le dispositif doit fournir des informations de routage devant être utilisées par les passerelles pour transmettre le datagramme à la destination et pour enregistrer les informations de chemin.

Les valeurs de IP-Option-length et IP-Option-Data sont spécifiées dans STD 0005 / RFC 791.

– Record Route - 0x07

Si cette option est présente, le dispositif doit également:

- envoyer des datagrammes IP émis avec cette option, fournissant des moyens d'enregistrer le chemin de ces datagrammes;
- comme routeur, envoyer des datagrammes IP acheminés avec l'option de chemin réglée en fonction de cette option.

Les valeurs de IP-Option-length et IP-Option-Data sont spécifiées dans STD 0005 / RFC 791.

– Internet Timestamp - 0x44

Si cette option est présente, le dispositif doit également:

- envoyer des datagrammes IP émis avec cette option, fournissant des moyens pour horodater le datagramme dans le chemin vers sa destination;
- comme routeur, envoyer des datagrammes IP acheminés avec

	l'option horodatage réglée en fonction de cette option. Les valeurs de IP-Option-length et IP-Option-Data values sont spécifiées dans STD 0005 / RFC 791.
subnet_mask	Contient le masque de sous-réseau. Lorsque le sous-réseautage est utilisé dans un segment de réseau, chaque dispositif concerné doit se comporter conformément aux règles de sous-réseautage. Pour ce faire, le dispositif, outre son adresse IP, a besoin de connaître aussi comment l'adresse IP est structurée au sein du segment mis en sous-réseau. L'attribut subnet_mask contient cette information. Avec IPv4, le subnet_mask est un mot de 32 bits, exprimé exactement dans le même format qu'une adresse IP (par exemple 255.255.255.0), mais a une autre signification: les bits '0' du subnet_mask indiquent la partie de l'adresse IP qui est encore utilisée comme Device_ID sur un réseau IP mis en sous-réseautage[7].
gateway_IP_address	Contient l'adresse IP du dispositif passerelle. Dans la plupart des mises en œuvre IP, il existe un code dans le module qui gère les datagrammes sortants pour décider si un datagramme peut être envoyé directement à la destination sur le réseau local ou s'il doit être envoyé vers une passerelle. Afin de pouvoir envoyer des datagrammes non locaux vers la passerelle, le dispositif doit connaître l'adresse IP du dispositif passerelle assignée au segment de réseau donné. Si aucune adresse IP n'est assignée, la valeur est 0.
use_DHCP_flag	Lorsque ce fanion est mis à TRUE, le dispositif utilise le protocole DHCP (protocole de configuration de serveur dynamique) pour déterminer dynamiquement les paramètres IP_address, subnet_mask et gateway_IP_address. D'autre part, lorsque le fanion est mis à FALSE, les paramètres IP_address, subnet_mask et gateway_IP_address doivent être réglés localement.
primary_DNS_ address	L'adresse IP du Serveur de noms de domaine (DNS) primaire. Si aucune adresse IP n'est assignée, la valeur est 0.
secondary_DNS_ address	L'adresse IP du Serveur de noms de domaine (DNS) secondaire. Si aucune adresse IP n'est assignée, la valeur est 0.

[7] Voir plus de détails concernant le sous-réseautage dans les documents RFC 940 et RFC 950.

Description de la méthode

add_mc_IP_address (IP_Address)	Ajoute une adresse IP de multidiffusion au tableau multicast_IP_Address. IP_Address::= double-long-unsigned
delete_mc_IP_ address (IP_Address)	Supprime une adresse IP du tableau multicast_IP_Address. L'adresse IP à effacer est identifiée par sa valeur. IP_Address::= double-long-unsigned
get_nbof_mc_IP_ addresses **(data)**	Retourne le nombre d'adresses IP contenues dans le tableau multicast_IP_Address. data::= unsigned

5.7.3 MAC address setup (class_id: 43, version: 0)

NOTE 1 Le nom et l'usage de cette classe d'interfaces ont été changés de "Ethernet setup" en "MAC address setup" pour permettre une utilisation plus générale, sans changer la version.

Les instances de cette IC contiennent l'adresse MAC du dispositif physique. (Ou, plus généralement, un dispositif ou un logiciel). Il doit y avoir une instance de cette IC pour chaque interface réseau d'un dispositif physique.

NOTE 2 Dans le cas du profil de communication basé sur HDLC à trois couches, l'adresse MAC (adresse HDLC inférieure) est contenue dans un objet "IEC HDLC setup" (configuration de HDLC CEI).

NOTE 3 Dans le cas du profil de communication S-FSK PLC, l'adresse MAC est contenue dans un objet "S-FSK Phy&MAC setup".

MAC address setup (configuration d'adresse MAC)	**0...n**	**class_id = 43, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				X
2. MAC_address	octet-string				x + 0x08
Méthodes spécifiques	***m/o***				

Description d'attribut

logical_name	Identifie l'instance de l'objet "MAC address setup" (Établissement d'adresse MAC). Voir 6.2.19.
MAC_address	Contient l'adresse MAC.

5.7.4 PPP setup (class_id: 44, version: 0)

Cette IC permet de modéliser l'établissement d'interfaces utilisant le protocole PPP, en gérant toutes les informations liées aux réglages de PPP associés à un dispositif physique donné et à une connexion de couche inférieure sur laquelle ces réglages sont utilisés. Il doit y avoir une instance de cette IC pour chaque interface réseau d'un dispositif physique.

PPP setup (établissement de PPP)		**0...n**	**class_id = 44, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. PHY_reference	(static)	octet-string				x + 0x08
3. LCP_options	(static)	LCP_options				x + 0x10
4. IPCP_options	(static)	IPCP_options				x + 0x18
5. PPP_authentication	(static)	PPP_auth				x + 0x20
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name

Identifie l'instance de l'objet “PPP setup” (Établissement de PPP). Voir 6.2.19.

PHY_reference

Référence un autre objet par son nom logique. L'objet référencé contient des informations relatives à l'interface de couche physique spécifique, prenant en charge la couche PPP.

LCP_options

Cet attribut contient les paramètres pour les options de Protocole de commande de liaison.

Ces options comprennent:

- Maximum-Receive-Unit (MRU, Type 1, STD 0051 / RFC 1661). Cette option de configuration peut être envoyée pour informer l'homologue que l'implémentation réalisée peut recevoir des paquets plus gros ou pour demander à l'homologue d'envoyer des paquets plus petits. La valeur par défaut est 1 500 octets;
- Async-Control-Character_Map (ACCM, Type 2, STD 0051 / RFC 1662): Cette option de configuration fournit une méthode pour négocier l'utilisation de la transparence de caractères de contrôle sur des liaisons asynchrones;
- Authentication-Protocol (Type 3, STD 0051 / RFC 1661, PAP, CHAP or EAP); Cette option de configuration fournit une méthode pour négocier l'utilisation d'un protocole spécifique pour l'authentification. Par défaut, l'authentification n'est pas exigée;
- Magic-Number (Type 5, STD 0051 / RFC 1661). Cette option de configuration fournit une méthode pour détecter des liaisons fonctionnant en boucle et autres anomalies de la couche liaison de données;
- Protocol-Field-Compression (PFC, Type 7, STD 0051 / RFC 1661). Cette option de configuration fournit une méthode pour négocier la compression du champ protocole PPP;
- Address-and-Control-Field-Compression (ACFC, Type 8, STD 0051 / RFC 1661). Cette option de configuration fournit une méthode pour négocier la compression des champs adresse et commande de couche liaison de données;
- FCS-Alternatives (Type 9, RFC 1570). Cette option de configuration fournit une méthode pour une mise en œuvre pour spécifier un autre format de FCS devant être envoyé par l'homologue ou pour négocier la FCS également;
- Call-back (Type 13, RFC 1570). Cette option de configuration fournit une méthode pour une mise en œuvre pour demander à un homologue joint de retourner l'appel. Cela procure une sécurité

renforcée en assurant que le site distant peut se connecter seulement à partir d'un seul emplacement tel que défini par le numéro de rappel.

La structure de cet attribut est la suivante:

```
LCP_options::= SEQUENCE OF LCP_options_element
LCP_options_element  ::= SEQUENCE
{
   LCP-option-type:     unsigned,
   LCP-option-length:   unsigned,
   LCP-option-data:     CHOICE
                        {
                        MRU                 [1] long-unsigned,
                        ACCM                [2] double-long-unsigned,
                        Auth-Prot           [3] long-unsigned,
                        Mag-Num             [5] double-long-unsigned,
                        ProtF-Compr         [7] boolean,
                        AdCtr-Compr         [8] boolean,
                        FCS-Alter           [9] unsigned,
                        Callback            [13] callback-data
                        }
}
LCP_option_type::= enum
{
        Max-Rec-Unit                        (1),
        Async-Control-Char-Map              (2),
        Auth-Protocol                       (3),
        Magic-Number                        (5),
        Protocol-Field-Compression          (7),
        Address-and-Ctr-Compression         (8),
        FCS-Alternatives                    (9),
        Callback                            (13)
}
```

Pour l'élément LCP-option-data, ce qui suit s'applique:

La valeur par défaut de MRU est 1 500.

La valeur de l'élément Auth-Prot (Authentication Protocol, protocole d'authentification) indique le protocole d'authentification utilisé sur la liaison PPP donnée.

Les valeurs possibles sont aujourd'hui:

0x0000 - Aucun protocole d'authentification n'est utilisé,
0xc023 - Le protocole PAP est utilisé,
0xc223 - Le protocole CHAP est utilisé,
0xc227 - Le protocole EAP est utilisé.

La valeur du champ "FCS-Alter (FCS Alternatives) options" identifie la FCS utilisée. Celles-ci sont attribuées comme suit:

bit 1 FCS Null,
bit 2 FCS 16 bits de CCITT,
bit 4 FCS 32 bits de CCITT

```
Callback-data::= SEQUENCE

{
```

```
        callback-active          boolean,    // par défaut: false,
        callback-data-length     unsigned,
        callback-operation       unsigned,
        callback-message         octet-string
}
```

Le membre callback-active indique si l'option de rappel est active sur cette liaison PPP.

callback-operation::= enum

```
{
        Location-is-determined-by-user-authentication        (0),
        Dialling-string                                      (1),
        Location-identifier                                  (2),
        E.164-number                                         (3),
        X500-distinguished-name                              (4),
        Location-is-determined-during-CBCP-negotiation       (6)
}
```

Le champ callback-message est zéro ou plus d'octets, et son contenu général est déterminé par le champ callback-operation. Le format réel de l'information est spécifique au site ou à l'application (Voir le document RFC 1570).

NOTE 1 Pour plus de détails sur le protocole de commande de liaison, voir le document RFC 1661.

NOTE 2 Pour les numéros assignés les plus récents, voir le document RFC 1700.

IPCP_options

Cet attribut contient les paramètres pour le protocole de commande IP – le module de protocole de commande de réseau du PPP pour négocier les paramètres IP sur les options de liaison PPP. Ces options comprennent:

- IP-Compression-Protocol (Type 2, RFC 1332). Ce paramètre indique le protocole de compression IP pris en charge au sein de la liaison PPP décrite par cet objet;
- Preferred-Local-IP-Address (Type 3, RFC 1332). Cette option de configuration fournit une façon de négocier l'adresse IP à utiliser sur l'extrémité locale de la liaison. Elle permet à l'expéditeur de la Configure-Request d'énoncer quelle adresse IP est souhaitée ou de demander que l'homologue fournisse les informations. L'homologue peut fournir cette information en acquittant négativement (NAK) l'option et en retournant une adresse IP valide;
- Preferred-Peer-IP-Addresses. Cette option de configuration fournit une façon de négocier l'adresse IP à utiliser sur l'extrémité distante de la liaison. Lorsque le paramètre Grant-Access-Only-to-Pref-Peer-on-List est mis à TRUE, le dispositif doit accepter la connexion PPP uniquement avec un dispositif distant ayant l'une des adresses IP figurant sur cette liste. Lorsque le paramètre Use-Static-IP-Pool est mis à TRUE, le dispositif Serveur COSEM doit tenter d'attribuer l'une de ces adresses IP au dispositif distant;
- Grant-Access-Only-to-Pref-Peer-on-List (GAO). Ce paramètre indique si, oui ou non, le dispositif peut accepter une connexion PPP uniquement avec des dispositifs homologues ayant une adresse IP figurant sur cette liste. Sa valeur par défaut est FALSE;
- Use-Static-IP-Pool (USIP). Ce paramètre indique si, oui ou non, il convient que le dispositif tente d'attribuer l'une des adresses IP des Preferred-Peer-IP-Addresses au dispositif situé à l'extrémité distante au cours de la phase de négociation d'adresses IP.

La structure de cet attribut est comme suit:

```
IPCP_options::= SEQUENCE OF IPCP_options_element

IPCP_options_element::= SEQUENCE
{
        IPCP-option-type        unsigned,
        IPCP-option-length      unsigned,
        IPCP-option-data        CHOICE
        {
        IP-Comp-Prot    [2] long-unsigned,
        Pref-Local-IP   [3] double-long-unsigned,
        Pref-Peer-IP    [20] SEQUENCE OF double-long-unsigned,
        GAO             [21] boolean,
        USIP            [22] boolean
        }
}
IPCP-option-type::= enum
{
        IP-Comp-Prot    (2),
        Pref-Local-IP   (3),
        Pref-Peer-IP    (20),
        GAO             (21),
        USIP            (22)
}
```

Les valeurs possibles pour le paramètre IP-Compression-Protocol (IP-Comp-Prot) sont aujourd'hui:

0x0000 – Aucune compression IP n'est utilisée (valeur par défaut),
0x002d – Van Jacobson (RFC 1332),
0x0061 – IP Header Compression (RFC 2507, 3544)
0x0003 – Robust Header Compression (ROHC, RFC 3241)

NOTE 3 Pour plus de détails sur IPCP, voir le document RFC 1332.

NOTE 4 Pour les numéros assignés les plus récents, voir le document RFC 1700.

PPP_ authentication

Contient les paramètres requis par la procédure d'authentification de PPP utilisée.

```
PPP_authentication::=  CHOICE
{
        No-authentication:      [0] NULL,
        PAP-login               [1] structure,
                                {
                                  user-name:        octet-string,
                                  PAP-password:             octet-string
                                }
        CHAP-algorithm          [2] structure,
                                {
                                  user-name:        octet-string,
                                  algorithm_id:             unsigned
                                  -- default: 5 (MD5)
                                }
        EAP-params:             [3] supported-EAP-types
}
```

Les valeurs possibles pour le paramètre CHAP-algorithm-id sont

aujourd'hui les suivantes:

0x05 – CHAP avec MD5 (par défaut),
0x06 – SHA-1,
0x80 – MS-CHAP,
0x81 – MS-CHAP-2

De nouvelles valeurs peuvent être utilisées à mesure qu'elles sont attribuées.

Lorsque CHAP est utilisé, un “secret” est également requis pour vérifier le “défi”[8] envoyé par le Client. Ce “secret” n'est pas accessible dans l'objet "PPP setup".

supported-EAP-types::= SEQUENCE

{
md5-challenge boolean,
one-time-password boolean,
generic-token-card boolean
}

5.7.5 GPRS modem setup (class_id: 45, version: 0)

Cette IC permet d'installer des modems GPRS en gérant toutes les données nécessaires à la gestion du modem.

GPRS modem setup (établissement de modem GPRS)	**0…n**	**class_id = 45, version = 0**			
Attributs	***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name (static)	octet-string				x
2. APN (static)	octet-string				x + 0x08
3. PIN_code (static)	long-unsigned				x + 0x10
4. quality_of_service (static)	structure				x + 0x18
Méthodes spécifiques	***m/o***				

Description d'attribut

logical_name Identifie l'instance de l'objet “GPRS modem setup” (Établissement de modem GPRS). Voir 6.2.19.

APN Définit le nom de point d'accès du réseau.

PIN_code Contient le numéro d'identification personnel.

quality_of_service Spécifie les paramètres de qualité de service. Il s'agit d'une structure de deux éléments:

- le premier élément définit les caractéristiques par défaut ou minimum du réseau concerné. Ces paramètres doivent être réglés à la valeur de meilleur effort;
- le second élément définit les paramètres demandés.

quality_of_service::= structure

8 Pour plus de détails relatifs à CHAP, voir le document RFC 1994.

```
{
        default:            qos_element,
        requested:  qos_element
}
qos_element::= structure
{
        precedence:             unsigned,
        delay:                  unsigned,
        reliability:            unsigned,
        peak throughput: unsigned,
        mean throughput:        unsigned
}
```

5.7.6 SMTP setup (class_id: 46, version: 0)

Cette IC permet d'établir un échange de données utilisant le protocole SMTP.

SMTP modem setup (installation de modem SMTP)		**0...n**	**class_id = 46, version = 0**			
Attributs		***Type de donnée***	***Min.***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. server_port	(static)	long-unsigned			25	x + 0x08
3. user_name	(static)	octet-string				x + 0x10
4. login_password	(static)	octet-string				x + 0x18
5. server_address	(static)	octet-string				x + 0x20
6. sender_address	(static)	octet-string				x + 0x28
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name Identifie l'instance de l'objet “SMTP setup” (Établissement de SMTP). Voir 6.2.19.

server_port Définit la valeur du port TCP-UDP lié à ce protocole. Par défaut, cette valeur est le numberID de port SMTP attribué par l'IANA:

smtp 25/tcp Simple Mail Transfer (transfert simple de courrier)

smtp 25/udp Simple Mail Transfer (transfert simple de courrier)

user_name Définit le nom d'utilisateur à utiliser pour la connexion au serveur SMTP.

login_password Mot de passe à utiliser pour la connexion. Si la chaîne est vide, cela signifie qu'il n'y a pas d'authentification.

server_address Définit l'adresse de serveur comme une chaîne d'octets. Cette adresse de serveur peut être un nom, qui doit pouvoir être résolu par le DNS primaire ou le DNS secondaire. S'il s'agit directement de l'adresse IP du serveur, qui est spécifiée ici, elle doit être une chaîne au format comportant des points.

EXEMPLE 163.187.45.87.

sender_address Définit l'adresse de l'expéditeur comme une chaîne d'octets. Cette adresse d'expéditeur peut être un nom. S'il s'agit directement de l'adresse IP de l'expéditeur, qui est spécifiée ici, elle sera une chaîne

au format comportant des points.

5.8 Classes d'interfaces pour l'établissement d'échange de données en utilisant le PLC à modulation S-FSK

5.8.1 Généralités

Ce paragraphe spécifie les classes d'interfaces COSEM pour établir et gérer les couches protocolaires des profils de communications DLMS/COSEM PLC:

- la couche physique S-FSK et la sous-couche MAC telles que définies dans la CEI 61334-5-1 et la CEI 61334-4-512

NOTE 1 Les IC pour établir d'autres profils de couches inférieures PLC peuvent être ajoutées ultérieurement.

- la sous-couche LLC telle que spécifiée dans la CEI 61334-4-32;
- la sous-couche LLC telle que spécifiée dans l'ISO/CEI 8802-2.

Les variables MIB variables / paramètres de liaison logique respectivement spécifiés dans la CEI 61334-4-512, CEI 61334-5-1, CEI 61334-4-32 et dans l'ISO/CEI 8802-2 ont été mis en correspondance avec les attributs et/ou méthodes des IC de la spécification COSEM. La spécification de ces éléments a été prise dans les normes citées ci-dessus et le texte a été adapté à l'environnement DLMS/COSEM.

NOTE 2 La CEI 61334-4-512 61334-4-512 spécifie également un certain nombre de variables de gestion à utiliser du côté Client. Cependant, le modèle d'objet du côté Client n'est pas couvert par la présente norme.

5.8.2 Définitions et abréviations relatives au profil PLC S-FSK

5.8.2.1 Définitions

Pour les besoins du présent document, les termes et définitions donnés dans les CEI/TR 62051 et CEI/TR 62051-1 s'appliquent, ainsi que les suivants.

5.8.2.1.1
initiateur
élément utilisateur d'une entité d'application de gestion de système (System Management Application Entity, SMAE) cliente

Note 1 à l'article: L'initiateur utilise le CIASE et le xDLMS ASE et est identifié par son titre de système.

[SOURCE: CEI 61334-4-511:2000, 3.8.1, modifiée]

5.8.2.1.2
initiateur actif
initiateur qui émet ou vient d'émettre une demande Register (recensement) CIASE lorsque le serveur est en état non configuré

[SOURCE: CEI 61334-4-511:2000, 3.9.1]

5.8.2.1.3
nouveau système
système serveur qui est dans l'état non configuré: son adresse MAC correspond à l'adresse NEW

[SOURCE: CEI 61334-4-511:2000, 3.9.3]

5.8.2.1.4
titre de nouveau système
titre de système d'un nouveau système

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

Note 1 à l'article: Il s'agit du titre de système dans le cas d'un système qui est dans le nouvel état.

5.8.2.1.5
système enregistré
système serveur ayant une adresse MAC individuelle valide (donc différente de l'adresse NEW, voir CEI 61334-5-1: Commande d'accès au support)

[SOURCE: CEI 61334-4-511:2000, 3.9.5, modifiée]

5.8.2.1.6
système de compte-rendu
système serveur qui émet un DiscoverReport

[SOURCE: CEI 61334-4-511:2000, 3.9.6, modifiée]

5.8.2.1.7
sous-intervalle
temps nécessaire pour émettre deux octets par la couche physique

Note 1 à l'article: Les intervalles de temps sont divisés en sous-intervalles dans le mode RepeaterCall de la couche physique.

5.8.2.1.8
intervalle de temps
temps nécessaire pour émettre une trame physique

Note 1 à l'article: Comme spécifié dans la CEI 61334-5-1:2001, 3.3.1, une trame physique comprend un préambule de 2 octets, un délimiteur de sous-trame de début de 2 octets, une PSDU de 38 octets et une pause de 3 octets.

5.8.2.2 Abréviations

Abréviation	Explication
CC	Current Credit (crédit courant, dans le cas du profil PLC S-FSK)
DC	Delta credit
IC	Initial credit (crédit initial)
CIASE	Configuration Initiation Application Service Element (élément de service d'application d'initiation de configuration)
FIFO	First-In-First-Out (premier entré, premier sorti)
LLC	Logical link control (commande de liaison logique)
MAC	Medium access control (Commande d'accès au support)
MIB	Management Information Base (base de données MIB)
UNC	Unconfigured, non configuré (serveur, dans le cas du profil S-FSK PLC)

5.8.3 Vue d'ensemble

Les objets COSEM pour établir le canal PLC et la couche LLC, si elle est mise en œuvre, doivent être situés dans le Dispositif logique de gestion des serveurs COSEM.

La Figure 15 montre un exemple avec un dispositif physique COSEM comprenant trois dispositifs logiques.

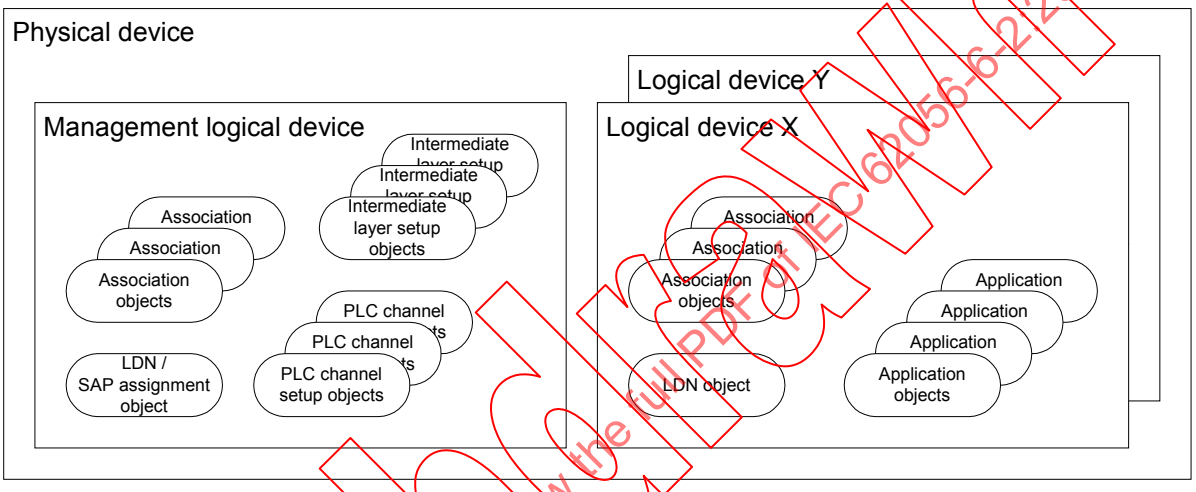
Chaque dispositif logique doit contenir un objet Logical Device Name (LDN, Nom de dispositif logique).

NOTE Comme dans l'exemple il y a plus d'un dispositif logique, le Dispositif logique de gestion obligatoire contient un objet "SAP Assignment" (Affectation de SAP) en lieu et place d'un objet "Logical Device Name" (Nom de dispositif logique).

Chaque dispositif logique contient un ou plusieurs objets Association, un par client pris en charge.

Le dispositif logique de gestion contient les objets d'établissement des couches physiques et MAC de la voie PLC, ainsi que des objets d'établissement pour la/les couche(s) intermédiaire(s). Il peut contenir des objets d'application supplémentaires.

Les autres dispositifs logiques contiennent, en plus des objets Association et Logical Device Name susmentionnés, d'autres objets d'applications, détenant des paramètres et des valeurs de mesure.



Légende

Anglais	Français
Physical device	Dispositif physique
Logical device	Dispositif logique
Management logical device	Dispositif logique de gestion
Association objects	Objets « Association »
Intermediate layer setup objects	Objets établissement de couche intermédiaire
LDN /SAP assignment objet	Objet d'affectation de SAP /LDN
PLC channel setup objects	Objets d'établissement de voie PLC
LDN object	Objet LDN
Application objects	Objets « objet d'application »

Figure 15 – Modèle d'objet des serveurs DLMS/COSEM

La CEI 61334-4-512 utilise des variables nommées selon DLMS pour modéliser les objets MIB et spécifie leur nom DLMS dans la plage 8…184. Pour la compatibilité avec des mises en œuvre existantes, les noms courts 8…400 [sic] sont réservés aux dispositifs utilisant les profils PLC S-FSK de la CEI 61334-5-1 sans COSEM. Par conséquent, cette plage ne doit pas être utilisée lors de la mise en correspondance des attributs et méthodes des objets COSEM spécifiés dans la présente norme avec les variables nommées selon DLMS (mise en correspondance par SN).

Tableau 10 – Mise en correspondance des variables MIB de la CEI 61334-4-512 avec les attributs/méthodes des IC de la COSEM

Nom	Référence (sauf indication contraire)	Classe d'interfaces	class_id / attribute / method
S-FSK Physical layer management (gestion de couche physique S-FSK)			
delta-electrical-phase	variable 1	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	50 / Attr. 3
max-receiving-gain	variable 2		50 / Attr. 4
max-transmitting-gain	–		50 / Attr. 5
search-initiator-threshold	–		50 / Attr. 6
frequencies	–		50 / Attr. 7
transmission-speed	–		50 / Attr. 15
MAC layer management (gestion de couche MAC)			
mac-address	variable 3	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	50 / Attr. 8
mac-group-addresses	variable 4		50 / Attr. 9
repeater	variable 5		50 / Attr. 10
repeater-status	–		50 / Attr. 11
search-initiator time-out	–	S-FSK MAC synchronization timeouts (class_id: 52, version: 0)	52 / Attr. 2
synchronization-confirmation-time-out	variable 6		52 / Attr. 3
time-out-not-addressed	variable 7		52 / Attr. 4
time-out-frame-not-OK	variable 8		52 / Attr. 5
min-delta-credit	variable 9	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	50 / Attr. 12
initiator-mac-address	CEI 61334-5-1:2001, 4.3.7.6		50 / Attr. 13
synchronization-locked	variable 10		50 / Attr. 14
IEC 61334-4-32 LLC layer management (gestion de couche LLC de la CEI 61334-4-32)			
max-frame-length	CEI 61334-4-32:1996, 5.1.4	IEC 61334-4-32 LLC setup (class_id: 55, version: 1)	55 / Attr. 2
reply-status-list	variable 11		55 / Attr. 3
broadcast-list	variable 12	–	–
L-SAP-list	variable 13	NOTE Dans DLMS/COSEM, les L-SAP des dispositifs logiques sont détenus par l'objet " SAP Assignment" (Affectation de SAP)	
ACSE management (gestion ACSE)			
application-context-list	variable 14	NOTE Dans DLMS/COSEM, les objets Association jouent un rôle similaire.	
Application management (gestion d'application)			
Active-initiator	variable 15	S-FSK Active initiator (class_id: 51, version: 0)	51 / Attr. 2
MIB system objects (objets système MIB)			
reporting-system-list	variable 16	S-FSK Reporting system list (class_id: 56, version: 0)	56 / Attr. 2
Other MIB objects (autres objets MIB)			
Reset-NEW-not-synchronized	variable 17	S-FSK Active initiator (class_id: 51, version: 0)	51 / Method 1
new-synchronization	CEI 61334-5-1:2001, 4.3.7.6	–	
initiator-electrical-phase	variable 18		50 / Attr. 2
broadcast-frames-counter	variable 19	S-FSK MAC counters (class_id: 53, version: 0)	53 / Attr. 4
repetitions-counter	variable 20		53 / Attr. 5
transmissions-counter	variable 21		53 / Attr. 6
CRC-OK-frames-counter	variable 22		53 / Attr. 7

Nom	Référence (sauf indication contraire)	Classe d'interfaces	class_id / attribute / method
CRC-NOK-frames-counter	–		53 / attr. 8
synchronization-register	variable 23		53 / Attr. 2
desynchronization-listing	variable 24		53 / Attr. 3

Le Tableau 10 ci-dessus montre la mise en correspondance des variables MIB avec les attributs et/ou méthodes des IC de la COSEM.

A noter que toutes les variables MIB spécifiées dans la CEI 61334-4-512 n'ont pas été mises en correspondance avec les attributs et méthodes des IC de COSEM. D'autre part, la présente norme spécifie un certain nombre de nouvelles variables de gestion.

5.8.4 S-FSK Phy&MAC set-up (class_id: 50, version: 1)

NOTE 1 L'utilisation de la version 0 de cette classe d'interface est déconseillée.

Une instance de la classe “S-FSK Phy&MAC set-up” stocke les données nécessaires pour établir et gérer la couche physique et la couche MAC du profil de couche inférieure PLC S-FSK.

S-FSK Phy&MAC setup (établissement de Phy et MAC S-FSK)		**0...n**	**class_id = 50, version = 1**			
Attributs		***Type de donnée***	***Min***	***Max.***	***Def.***	**Nom court**
1. logical_name	(static)	octet-string				x
2. initiator_electrical_phase	(static)	enum	0	3		x + 0x08
3. delta_electrical_phase	(dyn.)	enum	0	6		x + 0x10
4. max_receiving_gain	(static)	unsigned				x + 0x18
5. max_transmitting_gain	(static)	unsigned				x + 0x20
6. search_initiator_threshold	(static)	unsigned			98	x + 0x28
7. frequencies	(static)	frequencies_type				x + 0x30
8. mac_address	(dyn.)	long-unsigned			FFE	x + 0x38
9. mac_group_addresses	(static)	array				x + 0x40
10. repeater	(static)	enum				x + 0x48
11. repeater_status	(dyn.)	boolean				x + 0x50
12. min_delta_credit	(dyn.)	unsigned				x + 0x58
13. initiator_mac_address	(dyn.)	long-unsigned				x + 0x60
14. synchronization_locked	(dyn.)	boolean				x + 0x68
15. transmission_speed	(static)	enum	0	6	3	x + 0x70
Méthodes spécifiques		***m/o***				

Description d'attribut

logical_name Identifie l'instance de l'objet “S-FSK Phy&MAC setup” (Établissement de Phy et MAC S-SFK). Voir 6.2.20

initiator_electrical_phase Contient la variable MIB "initiator-electrical-phase" (variable 18) spécifiée dans la CEI 61334-4-512:2001, 5.8.

Il est inscrit par le système client pour indiquer la phase à laquelle

	il est connecté. enum: (0) Non définie (par défaut), (1) Phase 1, (2) Phase 2, (3) Phase 3 NOTE 2 Cette énumération diffère de celle de la CEI 61334-4-512.
delta_electrical_phase	Contient la variable MIB "delta-electrical-phase" (variable 1) spécifiée dans la CEI 61334-4-512:2001, 5.2 et la CEI 61334-5-1:2001, 3.5.5.3. Il indique la différence de phases entre la phase de connexion du client et la phase de connexion du serveur. Les valeurs suivantes sont prédéfinies: enum: (0) Non définie: le serveur est temporairement incapable de déterminer la différence de phases, (1) Le système serveur est connecté à la même phase que le système client. La différence de phases entre la phase de connexion du serveur et la phase de connexion du client est égale à: (2) 60 degrés, (3) 120 degrés, (4) 180 degrés, (5) -120 degrés, (6) -60 degrés,
max_receiving_gain	Contient la variable MIB "max-receiving-gain " (variable 2) spécifiée dans la CEI 61334-4-512:2001, 5.2 et la CEI 61334-5-1:2001, 3.5.5.3. Correspond au gain autorisé maximal destiné à être utilisé par le système serveur dans le mode réception. L'unité par défaut est dB. NOTE 3 La CEI 61334-4-512 ne spécifie pas d'unités. Les valeurs possibles du gain peuvent dépendre du matériel. Par conséquent, après écriture d'une valeur dans cet attribut, il convient de relire la valeur pour connaître la valeur exacte.
max_transmitting_gain	Contient la valeur du max-transmitting-gain. Correspond à l'affaiblissement maximal destiné à être utilisé par le système serveur dans le mode émission. L'unité par défaut est dB. Les valeurs possibles du gain peuvent dépendre du matériel. Par conséquent, après écriture d'une valeur dans cet attribut, il convient de relire la valeur pour connaître la valeur exacte.
search_initiator_ threshold	Cet attribut est utilisé dans le processus d'initiateur de recherche intelligente. Si la valeur du signal de l'initiateur se situe au-dessus de la valeur de cet attribut, un processus de synchronisation rapide est possible.

	La valeur par défaut est 98 dBµV.

frequencies	Contient les fréquences requises par la modulation S-FSK.
	
	frequencies_type::= structure
	
	{
	mark_frequency: double-long-unsigned,
	space_frequency: double-long-unsigned
	}
	L'unité par défaut est Hz.

mac_address	Contient la variable MIB "mac-address " (variable 3) spécifiée dans la CEI 61334-4-512:2001, 5.3 et la CEI 61334-5-1:2001, 4.3.7.6. NOTE 4 Les adresses MAC sont exprimées sur 12 bits. Contient la valeur de l'adresse de la jonction physique (adresse MAC) associée au système local. Dans l'état non configuré, l'adresse MAC est “NEW-address”. L'attribut est écrit localement par le CIASE lorsque le système est enregistré (au moyen d'un service Register). La valeur est utilisée dans chaque trame sortante ou entrante. La valeur par défaut est "NEW-address". Cet attribut est mis à NEW: • par la sous-couche MAC, une fois que le délai time-out-not-addressed est dépassé; • lorsqu'un système client "réinitialise" le système serveur. Voir 5.8.5. Lorsque cet attribut est mis à NEW: • le système perd sa synchronisation (fonction de la sous-couche MAC) • l'attribut mac_group_address est réinitialisé (tableau d'éléments 0); • le système libère automatiquement toutes les AA qui peuvent être libérées. NOTE 5 Le second élément n'est pas présent dans la CEI 61334-4-512. Les adresses MAC prédéfinies sont montrées au Tableau 11.
mac_group_addresses	Contient la variable MIB "mac-group-address " (variable 4) spécifiée dans la CEI 61334-4-512:2001, 5.3 et la CEI 61334-5-1:2001, 4.3.7.6. Contient un jeu d'adresses de groupe MAC utilisées à des fins de diffusion. array mac-address mac-address::= long-unsigned

	Les adresses ALL-configured-address, ALL-physical-address et NO-BODY ne sont pas incluses dans cette liste. Celles-ci sont des valeurs prédéfinies internes. Cet attribut doit être écrit par l'initiateur utilisant les services DLMS pour déclarer des adresses de groupe MAC spécifiques sur un système serveur. Cet attribut est lu localement par la sous-couche MAC lors de la vérification du champ adresse de destination d'une trame MAC non reconnue comme adresse individuelle ou comme l'une des trois valeurs prédéfinies (ALL-configured-address, ALL-physical-address et NO-BODY).
repeater	Contient la variable MIB "repeater" (variable 5) spécifiée dans la CEI 61334-4-512:2001, 5.3 et la CEI 61334-5-1:2001, 4.3.7.6. Il spécifie si, oui ou non, le système serveur répète effectivement toutes les trames. enum: (0) never repeater (ne répète jamais), (1) always repeater (toujours répéteur), (2) dynamic repeater (répéteur dynamique) Si la variable "repeater" est égale à 0, il convient que le système serveur ne répète jamais les trames. Si elle est mise à 1, le système serveur est un répéteur: il doit répéter toutes les trames reçues sans erreur et avec un crédit courant supérieur à zéro. Si elle est mise à 2, l'état statut de répéteur peut être modifié de façon dynamique par le serveur lui-même. NOTE 6 La valeur 2 n'est pas spécifiée dans la CEI 61334-4-512. Cet attribut est lu en interne par la sous-couche MAC chaque fois qu'une trame est reçue. La valeur par défaut doit être spécifiée dans des spécifications d'accompagnement.
repeater_status	Contient l'état courant de répéteur du dispositif. boolean: FALSE = pas de répéteur, TRUE = répéteur
min_delta_credit	Contient la variable MIB "min-delta-credit " (variable 9) spécifiée dans la CEI 61334-4-512:2001, 5.3 et la CEI 61334-5-1:2001, 4.3.7.6. NOTE 7 Seuls les trois bits de poids faible sont utilisés. Le Delta Credit (DC) est la soustraction des champs Initial Credit (IC) et Current Credit (CC) d'une trame MAC reçue correcte. La valeur minimum de delta-credit pour une trame MAC reçue correcte, dirigée vers un système serveur, est contenue dans cette variable.

	La valeur par défaut est mise au crédit initial maximal (voir CEI 61334-5-1:2001, 4.2.3.1 pour de plus amples explications concernant le crédit et la valeur de MAX_INITIAL_CREDIT). Un système client peut réinitialiser cette variable en mettant sa valeur au crédit initial maximal.
initiator_mac_address	Contient la variable MIB 'initiator-mac-address" spécifiée dans la CEI 61334-5-1:2001, 4.3.7.6. Sa valeur est soit l'adresse MAC de l'active-initiator, soit l'adresse NO-BODY, cela dépendant de la valeur de l'attribut synchronization_locked (voir ci-dessous). Voir aussi la CEI 61334-5-1:2001, 3.5.3, 4.1.6.3 et 4.1.7.2.
synchronization_locked	Contient la variable MIB "synchronization-locked " (variable 10) spécifiée dans la CEI 61334-4-512:2001, 5.3. Commande l'état locked/unlocked (verrouillé/déverrouillé) de la synchronisation. Voir CEI 61334-5-1 pour plus de détails. Si la valeur de cet attribut est égale à TRUE, le système est dans l'état synchronization-locked. Dans cet état, l'initiator-mac-address est toujours égale au champ adresse MAC de l'objet MIB "active-initiator MIB". Voir l'attribut 2 de l'IC "S-FSK Active initiator" en 5.8.5. Si la valeur de cet attribut est égale à FALSE, le système est dans l'état synchronization-unlocked. Dans cet état, l'attribut initiator_mac_address est toujours mis à la valeur NO-BODY: un changement de valeur dans le champ adresse MAC de l'objet MIB "active-initiator " n'a pas d'incidence sur le contenu de l'attribut "initiator_mac_address", qui reste à la valeur NO-BODY. La valeur par défaut de cette variable doit être spécifiée dans les spécifications de mise en œuvre. NOTE 8 Dans l'état synchronization-unlocked ("synchronisation déverrouillée), le serveur se synchronise sur n'importe quelle trame valide. Dans l'état "synchronisation verrouillée", le serveur se synchronise seulement sur les trames produites ou dirigées vers le système client dont l'adresse MAC est égale à la valeur de l'attribut initiator_mac_address.
transmission_speed	La vitesse de transmission prise en charge par le dispositif physique. Voir aussi CEI 61334-5-1:2001, 3.2.2. enum: (0) 50Hz: 300 bauds; 60 Hz: 360 bauds (1) 50Hz: 600 bauds; 60 Hz: 720 bauds (2) 50Hz: 1 200 bauds; 60 Hz: 1 440 bauds (3) -- par défaut 50Hz: 2 400 baud; 60 Hz: 2 880 bauds (4) 50Hz: 4 800 bauds; 60 Hz: 5 760 bauds (5) 50Hz: 7 200 bauds; 60 Hz: 8 640 bauds (6) 50Hz: 9 600 bauds; 60 Hz: 11 520 bauds