Green Book Ed. 9

Excerpt from

Companion Specification for Energy Metering

DLMS/COSEM Architecture and Protocols

DLMS User Association

dlms™

device
language
message
specification

CONTENTS

Foreword to Excerpt

This excerpt has been abstracted from the full technical report to give potential users a flavour of the content of the Green Book. It is not intended to provide sufficient information to allow a developer to implement the Protocol. This excerpt is about 27% of the content of the full Technical Report. Figure, Table and footnote numbering differs between this document and the full document.

Foreword

Copyright

This Edition 9 of the Green Book has a significant clarification in the operation of the General Block Transfer.

Some clarifications have been made, including clarifications with respect to the use of Attr. 0, system title with HLS

In addition, some editorial changes have been made. See the List of main changes.

This Technical Report is confidential. It may not be copied, nor handed over to persons outside the standardisation environment.

The copyright is enforced by national and international law. The "Berne Convention for the Protection of Literary and Artistic Works" which is signed by 173 countries worldwide and other treaties apply.

Liability

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

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

Acknowledgement

This document has been established by the WG Maintenance of the DLMS UA.

Clause 9.2, Information security in DLMS/COSEM is based on parts of NIST documents. Reprinted courtesy of the National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce. Not copyrightable in the United States.

Status of standardization

The content of Green Book Edition 8.2 is in line with IEC 62056-5-3:2017 Ed. 3.0, Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: DLMS/COSEM application layer.

To bring the changes in Green Book Editions 8.3 & 9 to international standardization, an update to IEC 62056-5-3:2017 will be initiated by IEC TC13 WG14.

List of main technical changes in Edition 9

Item	Clause	Change
1.	9.1.4.4.5	Substantively replaced for clarity. Description of Block Transfer mechanisms
2.	9.1.4.4.9	Substantively replaced for clarity. Description of General Block Transfer.
3.	9.3.2	Inclusion of system title for HLS mechanisms as needed in Calling_AP_Title
4.	9.3.7	Clarification regarding response to SET.request with Attr.0
5.	9.4.6.4	Addition/Correction of response APDU in table 79
6.	9.4.6.13.2	New content clarifying the procedure of operation of the General Block Transfer (GBT)
7.	9.4.6.13.5	New content clarifying the APDU processing sub-procedure of the General Block Transfer (GBT)

1 Scope

The DLMS/COSEM specification specifies an interface model and communication protocols for data exchange with metering equipment.

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

Communication protocols define how the data can be accessed and transported.

The DLMS/COSEM specification follows a three-step approach as illustrated in Figure 1:

- Step 1, Modelling: This covers the interface model of metering equipment and rules for data identification;
- Step 2, Messaging: This covers the services for mapping the interface model to protocol data units (APDU) and the encoding of this APDUs.
- Step 3, Transporting: This covers the transportation of the messages through the communication channel.

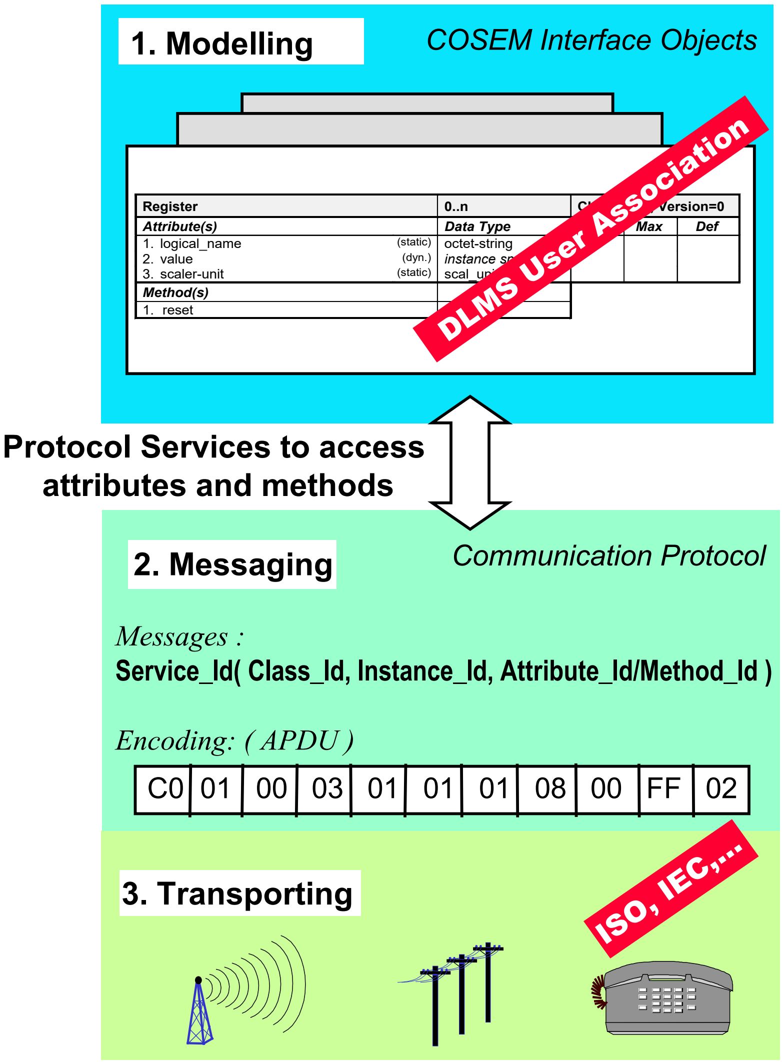


Figure 1 – The three steps approach of COSEM: Modelling – Messaging – Transporting

Step 1 is specified in the document "COSEM interface classes and the OBIS identification system" DLMS UA 1000-1. It specifies the COSEM interface classes, the OBIS identification system used to identify instances of these classes, called interface objects, and the use of interface objects for modelling the various functions of the meter.

Step 2 and 3 are specified in this Technical Report.

The DLMS/COSEM application layer (AL) specifies the services to establish logical connections between a client and (a) server(s) and the services to access attributes and methods of the COSEM objects. The DLMS/COSEM AL is specified in Clause 9.

DLMS/COSEM communication media specific profiles specify how application layer messages can be transported over various communication media. Each communication profile specifies the set of the protocol layers required to support the DLMS/COSEM AL on top. See also 4.8.

Large scale deployment of smart metering systems requires strong information security mechanisms to protect the privacy of energy consumers, the business interests of the energy and service providers and the security of the energy infrastructure.

DLMS/COSEM provides built-in security mechanisms from the outset. Initially, it provided mechanisms for the identification and authentication of clients and servers, as well as specific access rights to COSEM object attributes and methods within application associations (AAs) established between a client and a server. Ciphered APDUs were also available to allow protecting the messages exchanged between clients and servers.

In the next step, the details of ciphering using symmetric key algorithms, providing authentication and encryption as well as key transport mechanisms have been specified.

This Technical Report edition contains updates mostly focussed on ensuring that the General Block Transfer is easier to understand. This GBT is becoming increasingly significant with the increase in narrow band data networks being applied for metering and other IoT applications.

Rules for conformance testing are specified in the document DLMS UA 1001-1 "DLMS/COSEM Conformance Test Process".

Terms are explained in Clause 3 and in DLMS UA 1002 "COSEM Glossary of Terms".

2 Referenced Documents

Ref.	Title
DLMS UA 1000-1 Ed. 13:2019	COSEM Interface Classes and OBIS Identification System, the "Blue Book"
DLMS UA 1000-1	COSEM Interface Classes and OBIS Identification System, the "Blue Book" NOTE This undated reference is used unless a specific clause needs to be referenced.
DLMS UA 1001-1	DLMS/COSEM Conformance test and certification process, the "Yellow Book"
DLMS UA 1002 Ed. 1.0:2003	COSEM Glossary of Terms, "White Book"
IEC 61334-4-1:1996	Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 1: Reference model of the communication system
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 protocol – Distribution line message specification
IEC 61334-4-511:2000	Distribution automation using distribution line carrier systems – Part 4-511: Data communication protocols – Systems management – CIASE protocol
IEC 61334-5-1:2001	Distribution automation using distribution line carrier systems – Part 5-1: Lower layer profiles – The spread frequency shift keying (S-FSK) profile
IEC 61334-6:2000	Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule
IEC 62056-1-0	Electricity metering data exchange – The DLMS/COSEM suite – Part 1 0: Smart metering standardisation framework
IEC 62056-21:2002	Electricity metering – Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange
ISO/IEC 7498-1:1994	Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model
ISO/IEC 8649 Ed. 2.0:1996	Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element NOTE This standard has been replaced by ISO/IEC 15953:1999
ISO/IEC 8650-1 Ed 2.0:1996	Information technology – Open systems interconnection – Connection-oriented protocol for the association control service element: Protocol specification NOTE This standard has been replaced by ISO/IEC 15954:1999
ISO/IEC 8802-2 Ed. 3.0:1998	Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control
ISO/IEC 8824:2008	Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation
ISO/IEC 8825-1:2015	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
ISO/IEC 15953:1999	Information technology — Open Systems Interconnection — Service definition for the Application Service Object Association Control Service Element NOTE This standard cancels and replaces ISO/IEC 8649:1996 and its Amd. 1:1997 and Amd. 2:1998, of which it constitutes a technical revision.
ISO/IEC 15954:1999	Information technology — Open Systems Interconnection — Connection-mode protocol for the Application Service Object Association Control Service Element NOTE This standard cancels and replaces ISO/IEC 8650-1:1999 and its Amd. 1:1997 and Amd. 2:1998, of which it constitutes a technical revision.
EN13757-1:2014	Communication system for and remote reading of meters – Part 1: Data exchange
EN 13757-2:2004	Communication system for and remote reading of meters – Part 2 : Physical and Link Layer, Twisted Pair Baseband (M-Bus)

EN 13757-3:2018	Communication systems for meters - Part 3: Application protocols
EN 13757-4:2013	Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)
EN 13757-5:2015	Communication system for and remote reading of meters – Part 5: Wireless relaying
EN 13757-6:2015	Communication system for meters – Part 6: Local Bus
EN 13757-7:2018	Communication systems for meters - Part 7: Transport and security services
ITU-T V.44: 2000	SERIES V: DATA COMMUNICATION OVER THE TELEPHONE NETWORK – Error control – V.44:2000, Data compression procedures
ITU-T X.211	SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY – Information technology – Open systems interconnection – Physical Service Definition
ITU-T X.509:2008	SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY – Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks
ITU-T X.693 (11/2008)	Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)
ITU-T X.693 Corrigendum 1(10/2011)	Information technology – ASN.1 encoding rules: XML Encoding Rules (XER) Technical Corrigendum 1
ITU-T X.694 (11/2008)	Information technology – ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1
ITU-T X.694 Corrigendum 1 (10/2011)	Information technology – ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1Technical Corrigendum 1
ANSI C12.21:1999	Protocol Specification for Telephone Modem Communication
FIPS PUB 180-4:2012	Secure Hash Standard (SHS)
FIPS PUB 186-4:2013	Digital Signature Standard (DSS)
FIPS PUB 197:2001	Advanced Encryption Standard (AES)
NIST SP 800-21:2005	Guideline for Implementing Cryptography in the Federal Government
NIST SP 800-32:2001	Introduction to Public Key Technology and the Federal PKI Infrastructure
NIST SP 800-38D:2007	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
NIST SP 800-56A Rev. 2: 2013	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
NIST SP 800-57:2012	Recommendation for Key Management – Part 1: General (Revision 3)
NSA1	Suite B Implementer’s Guide to FIPS 186-3 (ECDSA), Feb 3rd 2010
NSA2	Suite B Implementer’s Guide to NIST SP800-56A, 28th July 2009
NSA3	NSA Suite B Base Certificate and CRL Profile, 27th May 2008
RFC 3394	Advanced Encryption Standard (AES) Key Wrap Algorithm, 2002, http://tools.ietf.org/html/rfc3394
RFC 4108	Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages, 2005, http://www.ietf.org/rfc/rfc4108
RFC 5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008, http://www.ietf.org/rfc/rfc5280
STD0005 (1981)	Internet Protocol. Also: RFC0791, RFC0792, RFC0919, RFC0922, RFC0950, RFC1112
STD0006 (1980)	User Datagram Protocol. Also: RFC0768
STD0007 (1981)	Transmission Control Protocol. Also: RFC0793

3 Terms, definitions and abbreviations and symbols

3.1 General DLMS/COSEM definitions

Term	Definition
ACSE APDU	APDU used by the Association Control Service Element (ACSE)
application association	cooperative relationship between two application entities, formed by their exchange of application protocol control information through their use of presentation services
application context	set of application service elements, related options and any other information necessary for the interworking of application entities in an application association
application entity	the system-independent application activities that are made available as application services to the application agent, e.g., a set of application service elements that together perform all or part of the communication aspects of an application process
application process	an element within a real open system which performs the information processing for a particular application [SOURCE: ISO/IEC 7498-1:1994, 4.1.4]
authentication mechanism	the specification of a specific set of authentication-function rules for defining, processing, and transferring authentication-values [SOURCE: ISO/IEC 15953:1999, 3.5.11]
block	one portion of an xDLMS APDU; the payload of a GBT APDU
client	application process running in the data collection system
client/server	relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfils the request
confirmed GBT procedure	procedure in which the sender sends streams of GBT APDUs and at the end of each stream the recipient acknowledges the blocks received and attempts recovering any missing blocks Note 1 to entry: A GBT stream consists of one or more GBT APDUs. Note 2 to entry: In the case of a confirmed GBT stream the end of the stream is indicated by the streaming bit to set to FALSE (0). In the case of an unconfirmed GBT stream the end of the stream is indicated by the Final bit set to TRUE (1).
COSEM	Companion Specification for Energy Metering; refers to the COSEM object model
COSEM APDU	comprises ACSE APDUs and xDLMS APDUs
COSEM data	COSEM object attribute values, method invocation and return parameters
COSEM interface class	entity with specific set of attributes and methods modelling a certain function on its own or in relation with other COSEM interface classes
COSEM object	instance of a COSEM interface class
DLMS/COSEM	refers to the application layer providing xDLMS services to access COSEM interface object attributes. Also refers to the DLMS/COSEM Application layer and the COSEM data model together.
DLMS context	a specification of the service elements of DLMS and semantics of communication to be used during the lifetime of an application association [SOURCE: IEC 61334-4-41:1996, 3.3.5]
entity authentication	corroboration that an entity is the one claimed [SOURCE: ISO/IEC 9798-1:2010, 3.14]
gap	empty space i.e. missing blocks in the receive queue RQ Note to entry: A receive queue RQ may have one or more gaps. In each gap, one or more blocks may be missing.
GBT APDU	xDLMS APDU with control information that carries a block of another xDLMS APDU or an empty block
GBT exchange	exchanging GBT APDUs that carry the service primitives of the same service
GBT stream	a sequence of GBT APDUs
general block transfer (GBT)	DLMS/COSEM application layer mechanism that can transfer any other xDLMS APDU that would be otherwise too long to fit into the maximum APDU size negotiated, in several blocks. Note to entry: GBT can be forced by including GBT parameters in the .request service primitive.

Term	Definition
logical device	abstract entity within a physical device, representing a subset of the functionality modelled with COSEM objects
master	central station – station which takes the initiative and controls the data flow
message	xDLMS APDU carrying a service primitive in an encoded form, which may also be cryptographically protected
mutual authentication	entity authentication which provides both entities with assurance of each other's identity [SOURCE: ISO/IEC 9798-1:2010, 3.18]. Note 1 to entry: The DLMS/COSEM HLS authentication mechanism provides mutual authentication.
overflow	more GBT APDUs received in one stream than the size of the GBT window
physical device	physical metering equipment, the highest level element used in the COSEM interface model of metering equipment
pull operation	style of communication where the request for a given transaction is initiated by the client
push operation	style of communication where the request for a given transaction is initiated by the server
receive queue (RQ)	placeholder for the blocks of the APDU received in a GBT stream
send queue (SQ)	placeholder for the blocks of the APDU to be sent
service-specific block transfer	DLMS/COSEM application layer mechanism that can transfer an xDLMS APDU corresponding to a specific service primitive, that would be otherwise too long to fit into the maximum APDU size negotiated, in several blocks
streaming window	number of GBT APDUs that can be received in a stream
system title	unique identifier of the system
server	an application process running in a metering equipment
slave	station responding to requests of a master station. Note 1 to entry: A meter is normally a slave station.
unconfirmed GBT procedure	procedure in which the sender sends and the recipient receives a single stream of GBT APDUs, the recipient does not acknowledge the blocks received and does not attempt to recover any blocks lost Note to entry: This is used to carry unconfirmed service requests from the client to the server or unsolicited service requests from the server to the client.
unilateral authentication	entity authentication which provides one entity with assurance of the other's identity but not vice versa [SOURCE: ISO/IEC 9798-1:2010, 3.39] Note 1 to entry: The DLMS/COSEM LLS authentication mechanism provides unilateral authentication.
xDLMS	extended DLMS; refers to the DLMS protocol with the extensions specified in this Technical Report
xDLMS APDU	APDU used by the xDLMS Application Service Element (xDLMS ASE)
xDLMS message	xDLMS APDU exchanged between a client and a server or between a third party and a server

3.2 Definitions related to cryptographic security

Term	Definitions *(source: NIST Special Publications)*
Access control	Restricts access to resources to only privileged entities. *Source: NIST SP 800-57:2012, Part 1*
Asymmetric key algorithm	See Public key cryptographic algorithm
Authentication	A process that establishes the source of information, provides assurance of an entity's identity or provides assurance of the integrity of communications sessions, messages, documents or stored data. *Source: NIST SP 800-57:2012, Part 1*

Term	Definitions *(source: NIST Special Publications)*
Authentication code	A cryptographic checksum based on an approved security function (also known as a Message Authentication Code) *Source: NIST SP 800-57:2012, Part 1*
Certificate	See public key certificate
Certification Authority (CA)	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing public key certificates and exacting compliance to a PKI policy *Source: NIST SP 800-56A Rev. 2: 2013*
Certificate Policy (CP)	A specialized form of administrative policy tuned to electronic transactions performed during certificate management. A Certificate Policy addresses all aspects associated with the generation, production, distribution, accounting, compromise recovery, and administration of digital certificates. Indirectly, a certificate policy can also govern the transactions conducted using a communications system protected by a certificate-based security system. By controlling critical certificate extensions, such policies and associated enforcement technology can support provision of the security services required by particular applications. *Source: NIST SP 800-32:2001*
Challenge	A time variant parameter generated by a verifier *Source: ITU-T X.811:1995, 3.8*
Ciphering	Authentication and / or encryption using symmetric key algorithms
Ciphertext	Data in its encrypted form *Source: NIST SP 800-57:2012, Part 1*
Cofactor	The order of the elliptic curve group divided by the (prime) order of the generator point (i.e. The base point) specified in the domain parameters *Source: NIST SP 800-56A Rev. 2: 2013*
Confidentiality	The property that sensitive information is not disclosed to unauthorized entities *Source: NIST SP 800-57:2012, Part 1*
Cryptographic algorithm	A well-defined computational procedure that takes variable inputs including a cryptographic key and produces an output *Source: NIST SP 800-57:2012, Part 1*
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot Examples include: 1. The transformation of plaintext data into ciphertext data, 2. The transformation of ciphertext data into plaintext data, 3. The computation of a digital signature from data, 4. The verification of a digital signature, 5. The computation of an authentication code from data, 6. The verification of an authentication code from data and a received authentication code, 7. The computation of a shared secret that is used to derive keying material. *Source: NIST SP 800-57:2012, Part 1*
Cryptoperiod	The time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect *Source: NIST SP 800-57:2012, Part 1*
Dedicated key	In DLMS/COSEM, a symmetric key used within a single instance of an Application Association. See also session key
Deprecated	Not recommended for new implementations
Digital signature	The result of a cryptographic transformation of data that, when properly implemented with supporting infrastructure and policy, provides the services of: 1. Origin authentication 2. Data integrity, and 3. Signer non-repudiation *Source: NIST SP 800-57:2012, Part 1*

Term	Definitions *(source: NIST Special Publications)*
Directly trusted CA	A directly trusted CA is a CA whose public key has been obtained and is being stored by an end entity in a secure, trusted manner, and whose public key is accepted by that end entity in the context of one or more applications *Source: ISO/IEC 15945:2002, 3.4*
Directly trusted CA key	A directly trusted CA key is a public key of a directly trusted CA. It has been obtained and is being stored by an end entity in a secure, trusted manner. It is used to verify certificates without being itself verified by means of a certificate created by another CA.Note 1 to entry: Directly trusted cas and directly trusted CA keys may vary from entity to entity. An entity may regard several cas as directly trusted cas. *Source: ISO/IEC 15945:2002, 3.5*
Distribution	See key distribution
Domain parameters	The parameters used with a cryptographic algorithm that are common to a domain of users *Source: NIST SP 800-56A Rev. 2: 2013*
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key *Source: NIST SP 800-57:2012, Part 1*
Ephemeral key	A cryptographic key that is generated for each execution of a key establishment process and that meets other requirements of the key type (e.g., unique to each message or session). In some cases ephemeral keys are used more than once, within a single "session (e.g., broadcast applications) where the sender generates only one ephemeral key pair per message and the private key is combined separately with each recipient's public key. *Source: NIST SP 800-57:2012, Part 1*
Global key	A key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a DLMS/COSEM Application Association, see also static symmetric key
Hash function	A function that maps a bit string of arbitrary length to a fixed-length bit string. Approved hash functions satisfy the following properties: 1) One-way: It is computationally infeasible to find any input that maps to any pre-specified output, and Collision resistant: It is computationally infeasible to find any two distinct inputs that map to the same output. *Source: NIST SP 800-57:2012, Part 1*
Hash value	The result of applying a hash function to information *Source: NIST SP 800-57:2012, Part 1*
Initialization vector (IV)	A vector used in defining the starting point of a cryptographic process *Source: NIST SP 800-57:2012, Part 1*
Identification	The process of verifying the identity of a user, process, or device, usually as a prerequisite for granting access to resources in an IT system *Source: NIST SP 800-47:2002*
Key	See cryptographic key
Key agreement	A (pair-wise) key-establishment procedure in which the resultant secret keying material is a function of information contributed by both participants, so that neither party can predetermine the value of the secret keying material independently from the contributions of the other party. Contrast with key-transport. *Source: NIST SP 800-56A Rev. 2: 2013*
Key-confirmation	A procedure to provide assurance to one party (the key-confirmation recipient) that another party (the key-confirmation provider) actually possesses the correct secret keying material and/or shared secret *Source: NIST SP 800-56A Rev. 2: 2013*
Key-derivation function	A function by which keying material is derived from a shared secret (or a key) and other information *Source: NIST SP 800-56A Rev. 2: 2013*
Key distribution	The transport of a key and other keying material from an entity that either owns the key or generates the key to another entity that is intended to use the key *Source: NIST SP 800-57:2012, Part 1*

Term	Definitions *(source: NIST Special Publications)*
Key-encrypting key	A cryptographic key that is used for the encryption or decryption of other keys. *Source: NIST SP 800-57:2012 Part 1* In DLMS/COSEM it is the master key.
Key establishment	The procedure that results in keying material that is shared among different parties *Source: NIST SP 800-56A Rev. 2: 2013*
Key pair	A public key and its corresponding private key; a key pair is used with a public key algorithm *Source: NIST SP 800-57:2012, Part 1*
Key revocation	A function in the lifecycle of keying material; a process whereby a notice is made available to affected entities that keying material should be removed from operational use prior to the end of the established cryptoperiod of that keying material *Source: NIST SP 800-57:2012, Part 1*
Key-transport	A (pair-wise) key-establishment procedure whereby one party (the sender) selects a value for the secret keying material and then securely distributes that value to another party (the receiver). Contrast with key agreement. *Source: NIST SP 800-56A Rev. 2: 2013*
Key wrapping	A method of encrypting keying material (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key *Source: NIST SP 800-57:2012, Part 1*
Message authentication code (MAC)	A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data *Source: NIST SP 800-57:2012, Part 1*
Message digest	The result of applying a hash function to a message. Also known as “hash value”. *Source: FIPS PUB 186-4:2013*
Named curve	A set of ECDH domain parameters is also known as a "curve". A curve is a "named curve" if the domain parameters are well known and defined and can be identified by an Object Identifier; otherwise, it is called a "custom curve". *Source: RFC 5349*
Nonce	A time-varying value that has at most an acceptably small chance of repeating. For example, the nonce may be a random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these. *Source: NIST SP 800-56A Rev. 2: 2013*
Non-repudiation	A service that is used to provide assurance of the integrity and origin of data in such a way that the integrity and origin can be verified by a third party as having originated from a specific entity in possession of the private key of the claimed signatory *Source: NIST SP 800-57:2012, Part 1*
Password	A string of characters (letters, numbers and other symbols) that are used to authenticate an identity or to verify access authorization or to derive cryptographic keys. *Source: NIST SP 800-57:2012, Part 1*
Plaintext	Intelligible data that has meaning and can be understood without the application of decryption *Source: NIST SP 800-57:2012, Part 1*
Private key	A cryptographic key, used with a public key cryptographic algorithm, which is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. Depending on the algorithm, the private key may be used, for example, to: 1) Compute the corresponding public key, 2) Compute a digital signature that may be verified by the corresponding public key, 3) Decrypt keys that were encrypted by the corresponding public key, or 4) Compute a shared secret during a key-agreement transaction. *Source: NIST SP 800-57:2012, Part 1*
Protected	Ciphered and /or digitally signed. Protection may be applied to xdlms apdus and/or to COSEM data.
Public key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. In an asymmetric (public) cryptosystem, the public key is associated with a private key. The public key may be known by anyone and, depending on the algorithm, may be used, for example, to:

Term	Definitions *(source: NIST Special Publications)*
	1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt keys that can be decrypted using the corresponding private key, or 3. Compute a shared secret during a key-agreement transaction. *Source: NIST SP 800-57:2012, Part 1*
Public-key certificate	A data structure that contains an entity's identifier(s), the entity's public key (including an indication of the associated set of domain parameters) and possibly other information, along with a signature on that data set that is generated by a trusted party, i.e. A certificate authority, thereby binding the public key to the included identifier(s). *Source: NIST SP 800-56A Rev. 2: 2013*
Public key (asymmetric) cryptographic algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible. *Source: NIST SP 800-57:2012, Part 1*
Public Key Infrastructure (PKI)	A framework that is established to issue, maintain and revoke public key certificates. *Source: NIST SP 800-57:2012, Part 1*
Receiver <key-transport>	The party that receives secret keying material via a key-transport transaction. Contrast with sender. *Source: NIST SP 800-56A Rev. 2: 2013*
Revoke a certificate	To prematurely end the operational period of a certificate effective at a specific date and time *Source: NIST SP 800-32:2001*
Root Certification Authority	In a hierarchical Public Key Infrastructure, the Certification Authority whose public key serves as the most trusted datum (i.e., the beginning of trust paths) for a security domain *Source: NIST SP 800-32:2001*
Secret key	A cryptographic key that is used with a secret key (symmetric) cryptographic algorithm that is uniquely associated with one or more entities and is not made public. The use of the term "secret" in this context does not imply a classification level, but rather implies the need to protect the key from disclosure *Source: NIST SP 800-57:2012, Part 1*
Security services	Mechanisms used to provide confidentiality, data integrity, authentication or non-repudiation of information *Source: NIST SP 800-57:2012, Part 1*
Security strength (also "bits of security")	A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system *Source: NIST SP 800-56A Rev. 2: 2013*
Self-signed certificate	A public key certificate whose digital signature may be verified by the public key contained within the certificate. The signature on a self-signed certificate protects the integrity of the data, but does not guarantee authenticity of the information. The trust of self-signed certificates is based on the secure procedures used to distribute them. *Source: NIST SP 800-57:2012, Part 1*
Sender <key-transport>	The party that sends secret keying material to the receiver in a key-transport transaction. Contrast with receiver. *Source: NIST SP 800-56A Rev. 2: 2013*
Session key	Cryptographic key established for use for a relatively short period of time. In DLMS/COSEM the dedicated key is a session key.
Shared secret	A secret value that has been computed using a key agreement scheme and is used as input to a key-derivation function/method. *Source: NIST SP 800-57:2012, Part 1*
Signature generation	Uses a digital signature algorithm and a private key to generate a digital signature on data. *Source: NIST SP 800-57:2012, Part 1*
Signature verification	Uses a digital signature algorithm and a public key to verify a digital signature on data *Source: NIST SP 800-57:2012, Part 1*
Signed data	Data upon which a digital signature has been computed

Term	Definitions *(source: NIST Special Publications)*
Static symmetric key	Key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a DLMS/COSEM Application Association Note In DLMS/COSEM it is known as global key.
Static key	A key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a cryptographic key establishment scheme. Contrast with an ephemeral key. *Source: NIST SP 800-57:2012, Part 1*
Subordinate Certification Authority	In a hierarchical PKI, a Certification Authority (CA) whose certificate signature key is certified by another CA, and whose activities are constrained by that other CA *Source: NIST SP 800-32:2001*
Symmetric key	A single cryptographic key that is used with a secret (symmetric) key algorithm *Source: NIST SP 800-57:2012, Part 1*
Symmetric key algorithm	A cryptographic algorithm that uses the same secret key for an operation and its complement (e.g., encryption and decryption) *Source: NIST SP 800-57:2012, Part 1*
Trust anchor	A public key and the name of a certification authority that is used to validate the first certificate in a sequence of certificates. The trust anchor public key is used to verify the signature on a certificate issued by a trust anchor certification authority. The security of the validation process depends upon the authenticity and integrity of the trust anchor. Trust anchors are often distributed as self-signed certificates. *Source: NIST SP 800-57:2012, Part 1*
Trusted party	A trusted party is a party that is trusted by an entity to faithfully perform certain services for that entity. An entity could be a trusted party for itself. *Source: NIST SP 800-56A Rev. 2: 2013*
Trusted third party	A third party, such as a CA, that is trusted by its clients to perform certain services. (By contrast, in a key establishment transaction, the participants, parties U and V, are considered to be the first and second parties.) *Source: NIST SP 800-56A Rev. 2: 2013*
X.509 certificate	The X.509 public-key certificate or the X.509 attribute certificate, as defined by the ISO/ITU-T X.509 standard. Most commonly (including in this document), an X.509 certificate refers to the X.509 public-key certificate. *Source: NIST SP 800-57:2012, Part 1*
X.509 public key certificate	A digital certificate containing a public key for entity and a name for the entity, together with some other information that is rendered unforgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard. *Source: NIST SP 800-57:2012, Part 1*

3.3 Definitions and abbreviations related to the Galois/Counter Mode

For more information see the complete Green Book.

3.4 General abbreviations

Abbreviation	Meaning
.cnf	.confirm service primitive
.ind	.indication service primitive
.req	.request service primitive
.res	.response service primitive

Abbreviation	Meaning
AA	Application Association
AARE	A-Associate Response – an APDU of the ACSE
AARQ	A-Associate Request – an APDU of the ACSE
ACPM	Association Control Protocol Machine
ACSE	Association Control Service Element
AE	Application Entity
AES	Advanced Encryption Standard
AL	Application Layer
AP	Application Process
APDU	Application Layer Protocol Data Unit
API	Application Programming Interface
ASE	Application Service Element
ASO	Application Service Object
ATM	Asynchronous Transfer Mode
A-XDR	Adapted Extended Data Representation
base_name	The short_name corresponding to the first attribute (“logical_name”) of a COSEM object
BER	Basic Encoding Rules
BD	Block Data
BN	Block Number
BNA	Block Number Acknowledged
BS	Bit string
BTS	Block Transfer Streaming
BTW	Block Transfer Window
CA	Certification Authority
CF	Control Function
CL	Connectionless
class_id	COSEM interface class identification code
CMP	Certificate Management Protocol. Refer to RFC 4210.
CO	Connection-oriented
COSEM	Companion Specification for Energy Metering
COSEM_on_IP	The TCP-UDP/IP based COSEM communication profile
CRC	Cyclic Redundancy Check
CRL	Certificate revocation list. Refer to RFC 5280.
CSR	Certificate Signing Request
DCE	Data Communication Equipment (communications interface or modem)
DCS	Data Collection System
DISC	Disconnect (a HDLC frame type)
DLMS	Device Language Message Specification
DM	Disconnected Mode (a HDLC frame type)
DSA	Digital Signature Algorithm specified in FIPS PUB 186-4:2013
DSAP	Data Link Service Access Point
DSO	Energy Distribution System Operator

Abbreviation	Meaning
DTE	Data Terminal Equipment (computers, terminals or printers)
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman key agreement protocol
ECDSA	Elliptic Curve Digital Signature Algorithm specified in ANSI X9.62 and FIPS PUB 186-4:2013
ECP	Elliptic Curve Point
EUI-64	64-bit Extended Unique Identifier
FCS	Frame Check Sequence
FDDI	Fibre Distributed Data Interface
FE	Field Element (in relation with public key algorithms)
FIPS	Federal Information Processing Standard
FRMR	Frame Reject (a HDLC frame type)
FTP	File Transfer Protocol
Gr	A GBT APDU received
GAK	Global Authentication Key
GBEK	Global Broadcast Encryption Key
GBT	General Block Transfer
GCM	Galois/Counter Mode (GCM), an algorithm for authenticated encryption with associated data
GMAC	A specialization of GCM for generating a message authentication code (MAC) on data that is not encrypted
GMT	Greenwich Mean Time
Gr.X	A field of a GBT APDU received
Gs	A GBT APDU sent
Gs.X	A field of a GBT APDU sent
GSM	Global System for Mobile communications
GUEK	Global Unicast Encryption Key
GW	Gateway
HCS	Header Check Sequence
HDLC	High-level Data Link Control
HES	Head End System, also known as Data Collection System NOTE The HES may be owned by the energy provider or the utility
HHU	Hand Held Unit
HLS	High Level Security (COSEM)
HMAC	Keyed-Hash Message Authentication Code specified in FIPS 198-1
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
I	Information (a HDLC frame type)
IANA	Internet Assigned Numbers Authority
IC	Interface Class
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
IV	Initialization Vector

Abbreviation	Meaning
KEK	Key Encrypting Key
LAN	Local Area Network
LB	Last Block
LDN	Logical Device Name
LLC	Logical Link Control (Sublayer)
LLS	Low Level Security
LNAP	Local Network Access Point
LPDU	LLC Protocol Data Unit
L-SAP	LLC sublayer Service Access Point
LSB	Least Significant Bit
LSDU	LLC Service Data Unit
m	mandatory, used in conjunction with attribute and method definitions
MAC	Medium Access Control (sublayer)
MAC	Message Authentication Code (cryptography)
MIB	Management Information Base
MSAP	MAC sublayer Service Access Point (in the HDLC based profile, it is equal to the HDLC address)
MSB	Most Significant Bit
MSC	Message Sequence Chart
MSDU	MAC Service Data Unit
N(R)	Receive sequence Number
N(S)	Send sequence Number
NDM	Normal Disconnected Mode
NIST	National Institute of Standards and Technology
NNAP	Neighbourhood Network Access Point
NRM	Normal Response Mode
o	optional, used in conjunction with attribute and method definitions
OBIS	Object Identification System
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OOB	Out of Band
OS	Octet string
OSI	Open System Interconnection
OTA	Over The Air
P/F	Poll/Final
PAR	Positive Acknowledgement with Retransmission
PDU	Protocol data unit
PhL	Physical Layer
PHSDU	PH SDU
PKCS	Public Key Cryptography Standard, established by RSA Laboratories
PKI	Public Key Infrastructure
PLC	Power line carrier
PPP	Point-to-Point Protocol

Abbreviation	Meaning
PSDU	Physical layer Service Data Unit
PSTN	Public Switched Telephone Network
RA	Registration Authority
RLRE	A-Release Response – an APDU of the ACSE
RLRQ	A-Release Request – an APDU of the ACSE
RNG	Random Number Generator
RNR	Receive Not Ready (a HDLC frame type)
RQ	Receive Queue
RR	Receive Ready (a HDLC frame type)
RSA	Algorithm developed by Rivest, Shamir and Adelman; specified in ANS X9.31 and PKCS #1.
S	A sequence of blocks in the RQ or SQ
SAP	Service Access Point
SDU	Service Data Unit
SHA	Secure Hash Algorithm; specified in FIPS PUB 180-4:2012
SNMP	Simple Network Management Protocol
SNRM	Set Normal Response Mode (a HDLC frame type)
SQ	Send Queue
STR	Streaming
tbsCertificate	To be signed certificate
TCP	Transmission Control Protocol
TDEA	Triple Data Encryption Algorithm
TL	Transport Layer
TPDU	Transport Layer Protocol Data Unit
TWA	Two Way Alternate
UA	Unnumbered Acknowledge (a HDLC frame type)
UDP	User Datagram Protocol
UI	Unnumbered Information (a HDLC frame type)
UNC	Unbalanced operation Normal response mode Class
USS	Unnumbered Send Status
V(R)	Receive state Variable
V(S)	Send state Variable
VAA	Virtual Application Association
WPDU	Wrapper Protocol Data Unit
xDLMS ASE	Extended DLMS Application Service Element
See also list of abbreviations specific to a cryptographic algorithm in the relevant clauses.	

3.5 Definitions, abbreviations, symbols and notation relevant for related to the Galois/Counter Mode

Term / Abbreviation	Symbol	Meaning
Additional authenticated data (AAD)	*A*	Input data to the authenticated encryption function that is authenticated but not encrypted
Authenticated decryption		Function of GCM in which the ciphertext is decrypted into the plaintext, and the authenticity of the ciphertext and the AAD are verified
Authenticated encryption		Function of GCM in which the plaintext is encrypted into the ciphertext and an authentication tag is generated on the AAD and the ciphertext
Authentication key	*AK*	A parameter that is part of the AAD
Block cipher		Parameterized family of permutations on bit strings of a fixed length; the parameter that determines the permutation is a bit string called the key
Ciphertext	*C*	Encrypted form of the plaintext
Encryption key,	*EK*	The block cipher key
Fixed field		In the deterministic construction of ivs, the field that identifies the device or context for the instance of the authenticated encryption function
Fresh		For a newly generated key, the property of being unequal to any previously used key
Gcm		Galois/counter mode
Invocation counter	*IC*	Part of the initialization vector. See also invocation field.
Invocation field		In the deterministic construction of ivs, the field that identifies the sets of inputs to the authenticated encryption function in a particular device or context
Initialization vector	*IV*	Nonce that is associated with an invocation of authenticated encryption on a particular plaintext and AAD
Key		Parameter of the block cipher that determines the selection of the forward cipher function from the family of permutations
Bit length	*len(X)*	The bit length of the bit string *X*.
Octet length	*LEN(X)*	The octet length of the octet string *X*.
Plaintext	*P*	Input data to the authenticated encryption function that is both authenticated and encrypted
Security control byte	*SC*	Byte that provides information on the ciphering applied
Security header	*SH*	Concatenation of the security control byte *SC* and the invocation counter: *SH* = *SC* II *IC.*
System title	*Sys-T*	A unique identifier of the system
Authentication tag	*T*	A cryptographic checksum on data that is designed to reveal both accidental errors and the intentional modification of the data.
Tag length	*t*	The bit length of the authentication tag. NOTE This is the same as len*(T)*
	X II Y	Concatenation of two strings, *X* and *Y*.

3.6 Definitions, abbreviations, symbols and notation relevant for the ECDSA algorithm

Symbol	Meaning
d	The ECDSA private key, which is an integer in the interval [1, n − 1].
$Q = (x_Q, y_Q)$	An ECDSA public key. The coordinates x_Q and y_Q are integers in the interval [0, q − 1], and $Q = dG$.
k	The ECDSA per-message secret number, which is an integer in the interval [1, n − 1].
r	One component of an ECDSA digital signature. It is an integer in [1, n − 1]. See the definition of (r, s).
s	One component of an ECDSA digital signature. It is an integer in [1, n − 1]. See the definition of (r, s).
(r, s)	An ECDSA digital signature, where r and s are the digital signature components.
M	The message that is signed using the digital signature algorithm.
Hash(M)	The result of a hash computation (message digest or hash value) on message M using an approved hash function.

3.7 Definitions, abbreviations, symbols and notation relevant for the key agreement algorithms

Symbol	Meaning
$d_{e,U}$, $d_{e,V}$	Party U's and Party V's ephemeral private keys. These are integers in the range [1, n-1].
$d_{s,U}$, $d_{s,V}$	Party U's and Party V's static private keys. These are integers in the range [1, n-1].
ID_U	The identifier of Party U (the initiator)
ID_V	The identifier of Party V (the responder)
$Q_{e,U}$, $Q_{e,V}$	Party U's and Party V's ephemeral public keys. These are points on the elliptic curve defined by the domain parameters.
$Q_{s,U}$, $Q_{s,V}$	Party U's and Party V's static public keys. These are points on the elliptic curve defined by the domain parameters.
U, V	Represent the two parties in a (pair-wise) key establishment scheme.
Z	A shared secret (represented as a byte string) that is used to derive secret keying material using a key derivation method. *Source: NIST SP 800-56A Rev. 2: 2013*

3.8 Abbreviations related to the DLMS/COSEM M-Bus communication profile

Abbrev	Term	Standard domain
ACC	Access number field	M-Bus
ALA	Application Layer Address	M-Bus
CFG	Configuration byte	M-Bus
CI_{ELL}	CI field introducing the extended link layer (wireless M-Bus)	M-Bus
CI Field	Control Information field	M-Bus
CI_{TL}	CI field introducing the transport layer	M-Bus
DTSAP	Destination Transport Service Access Point	Telecontrol
ELL	Extended Link Layer	M-Bus
ELLA	Extended Link Layer Address	M-Bus
FIN (bit)	Final Bit	Telecontrol
FT1.2	Data Integrity Format class FT1.2	Telecontrol

Abbrev	Term	Standard domain
FT3	Data Integrity Format Class FT3	Telecontrol
LLA	Link Layer Address	M-Bus
STS	Status byte	M-Bus
STSAP	Source Transport Service Access Point	Telecontrol
wM-Bus	Wireless M-Bus	M-Bus

4 Information exchange in DLMS/COSEM

4.1 General

This Clause 4 introduces the main concepts of information exchange in DLMS/COSEM.

The objective of DLMS/COSEM is to specify a standard for a business domain oriented interface object model for metering devices and systems, as well as services to access the objects. Communication profiles to transport the messages through various communication media are also specified.

The term "metering devices" is an abstraction; consequently “metering device” may be any type of device for which this abstraction is suitable.

The COSEM object model is specified in DLMS UA 1000-1, the “Blue Book”. The COSEM objects provide a view of the functionality of metering devices through their communication interfaces.

This Technical report, the “Green Book” specifies the DLMS/COSEM application layer, lower protocol layers and communication profiles.

The key characteristics of data exchange using DLMS/COSEM are the following:

- metering devices can be accessed by various parties: clients and third parties;
- mechanisms to control access to the resources of the metering device are provided; these mechanisms are made available by the DLMS/COSEM AL and the COSEM objects (“Association SN / LN” object, “Security setup” object);
- security and privacy is ensured by applying cryptographical protection to xDLMS messages and to COSEM data;
- low overhead and efficiency is ensured by various mechanisms including selective access, compact encoding and compression;
- at a metering site, there may be single or multiple metering devices. In the case of multiple metering devices at a metering site, a single access point can be made available;
- data exchange may take place either remotely or locally. Depending on the capabilities of the metering device, local and remote data exchange may be performed simultaneously without interfering with each other;
- various communication media can be used on local networks (LN), neighbourhood networks (NN) and wide area networks (WAN).

The key element to ensure that the above requirements are met is the Application Association (AA) – determining the contexts of the data exchange – provided by the DLMS/COSEM AL. For details, see the relevant clauses below.

4.2 Communication model

DLMS/COSEM uses the concepts of the Open Systems Interconnection (OSI) model to model information exchange between meters and data collection systems.

NOTE Information in this context comprises xDLMS messages and COSEM data.

Concepts, names and terminology used below relate to the OSI reference model described in ISO/IEC 7498-1:1994. Their use is outlined in this clause and further developed in other clauses.

Application functions of metering devices and data collection systems are modelled by application processes (APs).

Communication between APs is modelled by communication between application entities (AEs). An AE represents the communication functions of an AP. There may be multiple sets of OSI communication functions in an AP, so a single AP may be represented by multiple AEs. However, each AE represents a single AP. An AE contains a set of communication capabilities called application service elements (ASEs). An ASE is a coherent set of integrated functions. These ASEs may be used independently or in combination. See also 9.1.2.

Data exchange between data collection systems and metering devices is based on the client/server model where data collection systems play the role of the client and metering devices play the role of the server. The client sends service requests to the server which sends service responses. In addition the server may initiate unsolicited service requests to inform the client about events or to send data on pre-configured conditions. See also 4.6.

In general, the client and the server APs are located in separate devices. Therefore, message exchange takes place via a protocol stack as shown in Figure 2.

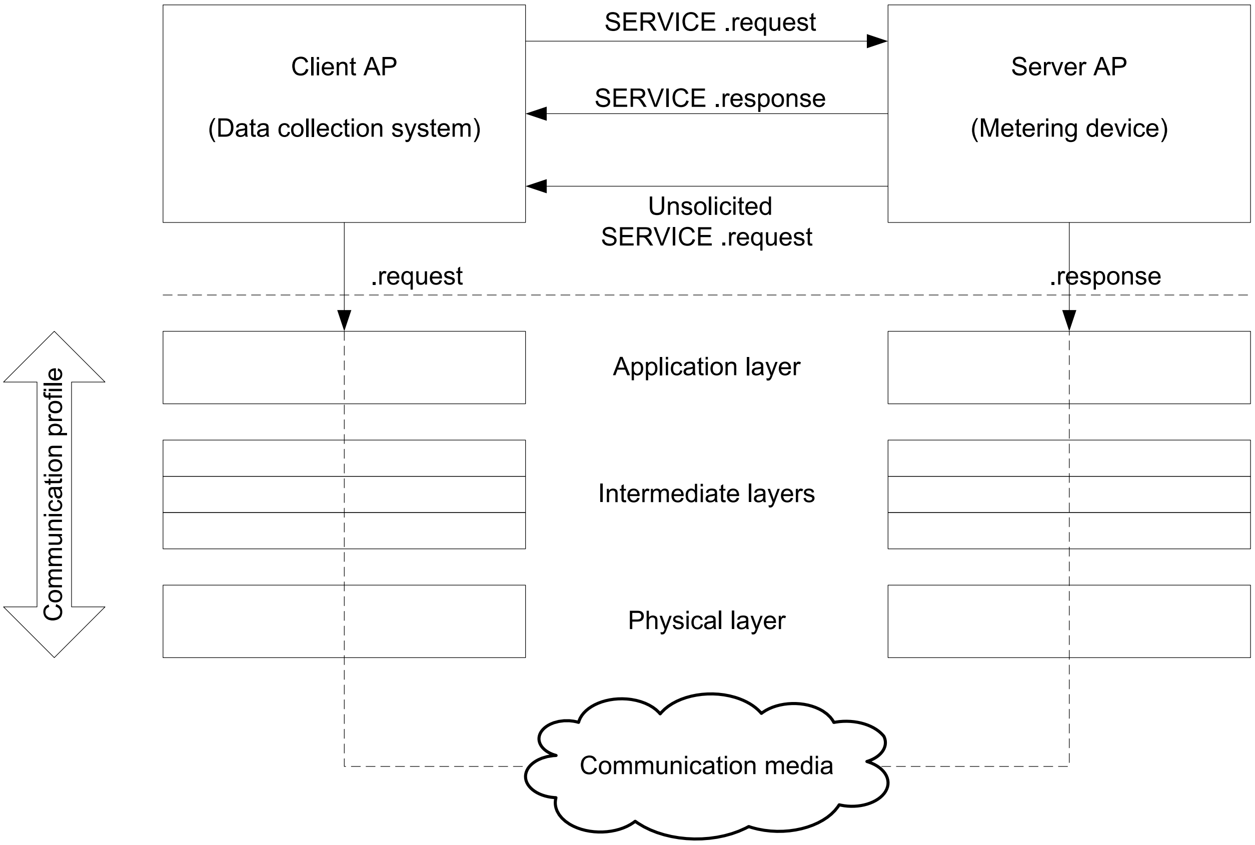


Figure 2 – Client–server model and communication protocols

4.3 Naming and addressing

4.3.1 General

Naming and addressing are important aspects in communication systems. A name identifies a communicating entity. An address identifies where that entity can be found. Names are mapped to addresses; this is known as the process of binding. Figure 3 shows the main elements of naming and addressing in DLMS/COSEM.

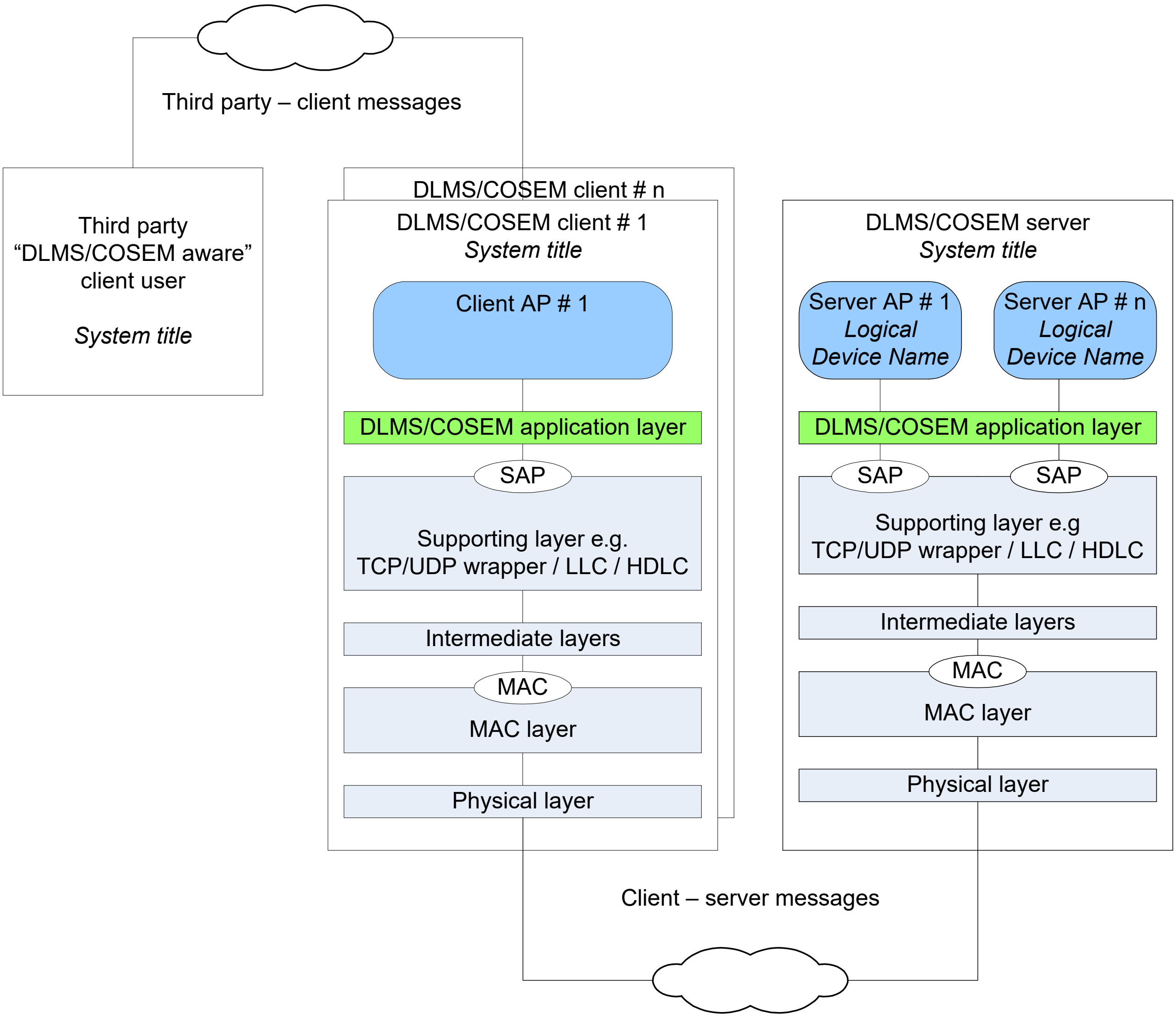


Figure 3 – Naming and addressing in DLMS/COSEM

4.3.2 Naming

DLMS/COSEM entities, including clients, servers as well as third party systems shall be uniquely named by their *system title*. System titles shall be permanently assigned.

Server physical devices may host one or more logical devices (LDs). LDs shall be uniquely identified by their Logical Device Name (LDN). LDs hosted by the same physical device share the system title. System titles are specified in 4.3.4. Logical device names are specified in 4.3.5.

4.3.3 Addressing

Each physical device shall have an appropriate address. It depends on the communication profile and may be a phone number, a MAC address, an IP network address or a combination of these.

NOTE For example, in the case of the 3-layer, connection-oriented, HDLC based communication profile, the lower HDLC address is the MAC address.

Physical device addresses may be pre-configured or may be assigned during a registration process, which also involves binding between the addresses and the system titles.

Each DLMS/COSEM client and each server – a COSEM logical device – is bound to a Service Access Point (SAP). The SAPs reside in the supporting layer of the DLMS/COSEM AL. Depending on the communication profile the SAP may be a TCP-UDP/IP wrapper address, an upper HDLC address, an LLC address etc. On the server side, this binding is modelled by the “SAP Assignment” IC; see DLMS UA 1000-1 Ed. 13:2019, 4.4.5.

The values of the SAPs on the client and the server side are specified in Table 1. The length of the SAPs depends on the communication profile.

Table 1 – Client and server SAPs

Client SAPs	
No-station	0x00
Client Management Process / CIASE [1]	0x01
Public Client	0x10
Open for client AP assignment	0x02 ...0x0F
	0x11 and up
Server SAPs	
No-station / CIASE [1]	0x00
Management Logical Device	0x01
Reserved for future use	0x02...0x0F
Open for server SAP assignment	0x10 and up
All-station (Broadcast)	Communication profile specific
[1] In the case of the DLMS/COSEM S-FSK PLC profile, see the complete Green Book.	
NOTE Depending on the supporting layer, the SAPs may be represented on one or more bytes.	

4.3.4 System title

The system title *Sys-T* shall uniquely identify each DLMS/COSEM entity that may be server, a client or a third party that can access servers via clients. The system title:

- shall be 8 octets long;
- shall be unique.

The leading (i.e., the 3 leftmost) octets should hold the three-letter manufacturer ID[1]. This is the same as the leading three octets of the Logical Device Name, see 4.3.5. The remaining 5 octets shall ensure uniqueness.

NOTE It can be derived for example from the last 12 digits of the manufacturing number, up to 999 999 999 999. This value converts to 0xE8D4A50FFF. Values above this, up to 0xFFFFFFFFFF (decimal 1 099 511 627 775) can also be used, but these values cannot be mapped to the last 12 digits of the manufacturing number.

Project specific companion specifications may specify a different structure. In that case, the details should be specified by the naming authority designated as such for the project.

The use of the system title in cryptographic protection of xDLMS messages and COSEM data is further specified in 9.2.3 and 9.2.7.

Before the cryptographic security algorithms can be used – this requires a ciphered application context – the peers have to exchange system titles. The following possibilities are available:

[1] Administered by the FLAG Association in co-operation with the DLMS UA.

- during the communication media specific registration process. For example, when the S-FSK PLC profile is used, system titles are exchanged during the registration process using the CIASE protocol; see the complete Green Book;
- in all communication profiles, system titles may be exchanged during AA establishment using the COSEM-OPEN service, see 9.3.2, carried the AARQ / AARE APDU. If the system titles sent / received during AA establishment are not the same as the ones exchanged during the registration process, the AA shall be rejected;
- by writing the *client_system_title* attribute and by reading the *server_system_title* attribute of “Security setup” objects, see DLMS UA 1000-1 Ed. 13:2019, 4.4.7.

In the case of broadcast communication, only the client sends the system title to the server.

4.3.5 Logical Device Name

Logical Device Name (LDN) shall be as specified in DLMS UA 1000-1 Ed. 13:2019, 4.1.8.2.

4.3.6 Client user identification

The client user identification mechanism allows a server to distinguish between different users on the client side and to log their activities accessing the meter. It is specified in DLMS UA 1000-1 Ed. 13:2019, 4.4.2. Naming of client users is outside the scope of this Technical Report.

4.4 Connection oriented operation

The DLMS/COSEM AL is connection oriented. See also 9.1.3.

A communication session consists of three phases, as it is shown in Figure 4:

- first, an application level connection, called Application Association (AA), is established between a client and a server AE; see also 9.1.3. Before initiating the establishment of an AA, the peer PhLs of the client and server side protocol stacks have to be connected. The intermediate layers may have to be connected or not. Each layer, which needs to be connected, may support one or more connections simultaneously;
- once the AA is established, message exchange can take place;
- at the end of the data exchange, the AA is released.

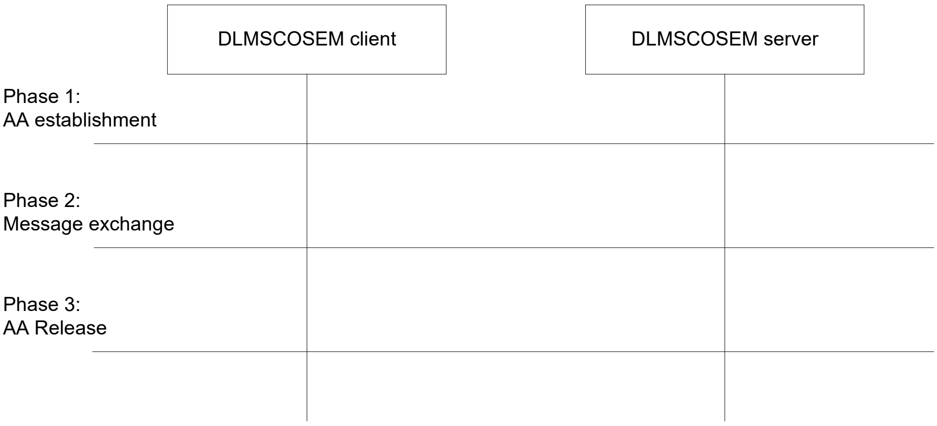


Figure 4 – A complete communication session in the CO environment

For the purposes of very simple devices, one-way communicating devices, and for multicasting and broadcasting pre-established AAs are also allowed. For such AAs the full communication session may include only the message exchange phase: it can be considered that the connection establishment phase has been already done somewhere in the past. Pre-established AAs cannot be released. See also the complete Green Book.

4.5 Application associations

4.5.1 General

Application Associations (AAs) are logical connections between a client and a server AE. AAs may be established on the request of a client using the services of the connection-oriented ACSE of the AL or may be pre-established. They may be confirmed or unconfirmed. See also 9.1.3.

NOTE 1 A pre-established AA can be considered to have been established in the past.

NOTE 2 Servers cannot initiate the establishment of an AA.

A COSEM logical device may support one or more AAs, each with a different client. Each AA determines the contexts in which information exchange takes place.

A confirmed AA is proposed by the client and accepted by the server provided that:

- the user of the client is known by the server, see 4.3.6;
- the application context proposed by the client – see 4.5.2 – is acceptable for the server;
- the authentication mechanism proposed by the client – see 4.5.3 – is acceptable for the server and the authentication is successful;
- the elements of the xDLMS context – see 4.5.4 – can be successfully negotiated between the client and the server.

An unconfirmed AA is also proposed by a client with the assumption that the server will accept it. No negotiation takes place. Unconfirmed AAs are useful for sending broadcast messages from the client to servers.

AAs are modelled by COSEM "Association SN / LN" objects that hold the SAPs identifying the associated partners, the name of the *application context*, the name of the *authentication mechanism*, and the *xDLMS context.*

The "Association SN / LN" objects also determine a specific set of access rights to COSEM object attributes and methods and they point to (reference) a "Security setup" object that hold the elements of the security context. The access rights and the security context may be different in each AA. These objects are specified in DLMS UA 1000-1.

4.5.2 Application context

The application context determines:

- the set of Application Service Elements (ASEs) present in the AL;
- the referencing style of COSEM object attributes and methods: short name (SN) referencing or logical name (LN) referencing. See also 9.1.4.3.1;
- the transfer syntax;
- whether ciphering is used or not.

Application contexts are identified by names, see 9.4.2.2.2.

4.5.3 Authentication

In communication systems entity authentication is a fundamentally important security service. The goal of entity authentication is to establish whether the claimant of a certain identity is in fact who it claims to be. In order to achieve this goal, there should be a pre-existing relation which links the entity to a secret.

In DLMS/COSEM, authentication takes place during AA establishment.

In confirmed AAs either the client (unilateral authentication) or both the client and the server (mutual authentication) can authenticate itself.

In an unconfirmed AA, only the client can authenticate itself.

In pre-established AAs, authentication of the communicating partners is not available.

Once the AA is established, COSEM object attributes and methods can be accessed using xDLMS services subject to the prevailing security context and access rights in the given AA.

The COSEM authentication mechanisms are specified in 9.2.2.2.2. The authentication mechanisms are identified by names, see 9.4.2.2.3.

4.5.4 xDLMS context

The xDLMS context determines the set of xDLMS services and capabilities that can be used in a given AA. See 9.1.4.

4.5.5 Security context

The security context is relevant when the application context stipulates ciphering. It comprises the security suite, the security policy, the security keys and other security material. See also the complete Green Book.It is managed by “Security setup” objects.

4.5.6 Access rights

Access rights determine the rights of the client(s) to access COSEM object attributes and methods within an AA. The set of access rights depend on the role of the client and is pre-configured in the server. See also the complete Green Book.

NOTE The roles and the related access rights are subject to project specific companion specifications. Examples for roles are meter reader, meter service / communication service / energy provider, manufacturer, end user etc.

4.6 Messaging patterns

The messaging patterns available between a DLMS/COSEM client and server are shown in Figure 5.

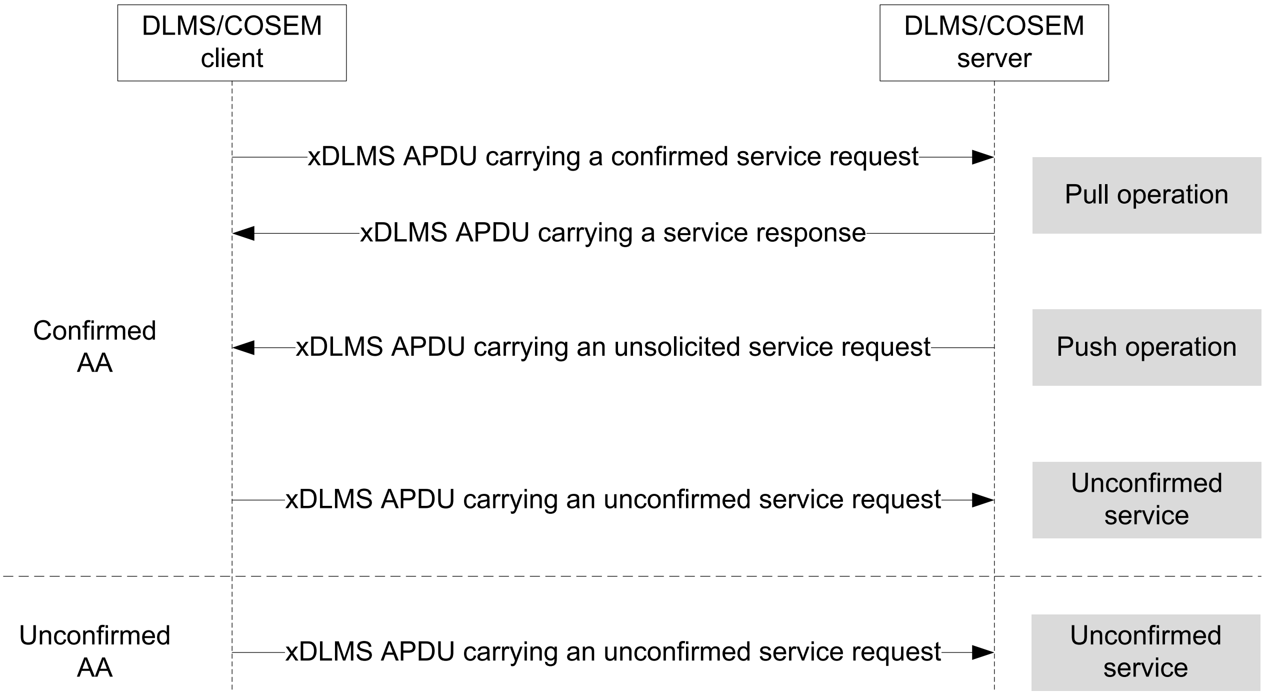

DLMS/COSEM client
DLMS/COSEM server
xDLMS APDU carrying a confirmed service request
xDLMS APDU carrying a service response
Pull operation
Confirmed AA
xDLMS APDU carrying an unsolicited service request
Push operation
xDLMS APDU carrying an unconfirmed service request
Unconfirmed service
Unconfirmed AA
xDLMS APDU carrying an unconfirmed service request
Unconfirmed service

Figure 5 – DLMS/COSEM messaging patterns

In confirmed AAs:

- the client can send confirmed service requests and the server responds: *pull operation*;
- the client can send unconfirmed service requests. The server does not respond;
- the server can send unsolicited service requests to the client: *push operation*.

  NOTE The unsolicited services may be InformationReport (with SN referencing), EventNotification (with LN referencing) or DataNotification (used both with SN and LN referencing).

In unconfirmed AAs:

- only the client can initiate service requests and only unconfirmed ones. The server cannot respond and it cannot initiate service requests.

4.7 Data exchange between third parties and DLMS/COSEM servers

Third parties – that are outside the DLMS/COSEM client-server relationship – may also exchange information with servers, using a client as a broker. To support end-to-end security, such third parties shall be “DLMS/COSEM aware” meaning that they shall be able to send messages to the client that contain properly formatted xDLMS APDUs carrying properly formatted COSEM data, and that they shall be able to process messages received from the server via the client. See also the complete Green Book.

Messages from the server to the third party may be solicited or unsolicited.

4.8 Communication profiles

Communication profiles specify how the DLMS/COSEM AL and the COSEM data model modelling the Application Process (AP) are supported by the lower, communication media specific protocol layers.

Communication profiles comprise a number of protocol layers. Each layer has a distinct task and provides services to its upper layer and uses services of its supporting layer(s). The client and server COSEM APs use the services of the highest protocol layer, that of the DLMS/COSEM AL. This is the only protocol layer containing COSEM specific element(s): the xDLMS ASE; see 9.1.4. It may be supported by any layer capable of providing the services required by the DLMS/COSEM AL. The number and type of lower layers depend on the communication media used.

A given set of protocol layers with the DLMS/COSEM AL and the COSEM object model on top constitutes a particular DLMS/COSEM communication profile. Each profile is characterized by the protocol layers included and their parameters.

Figure 6 shows a generic DLMS/COSEM communication profile, including:

- the COSEM object model modelling the Application Process. For each communication media, media-specific setup interface classes are specified;
- the DLMS/COSEM application layer;
- the DLMS/COSEM transport layer, present in internet capable profiles;
- the convergence layers that bind the MAC layer to the DLMS/COSEM AL either directly or through the DLMS/COSEM transport layer;
- the media specific physical and MAC layers; and
- the connection managers.

A single physical device may support more than one communication profile to allow data exchange using various communication media. In such cases it is the task of the client side AP to decide which communication profile should be used.

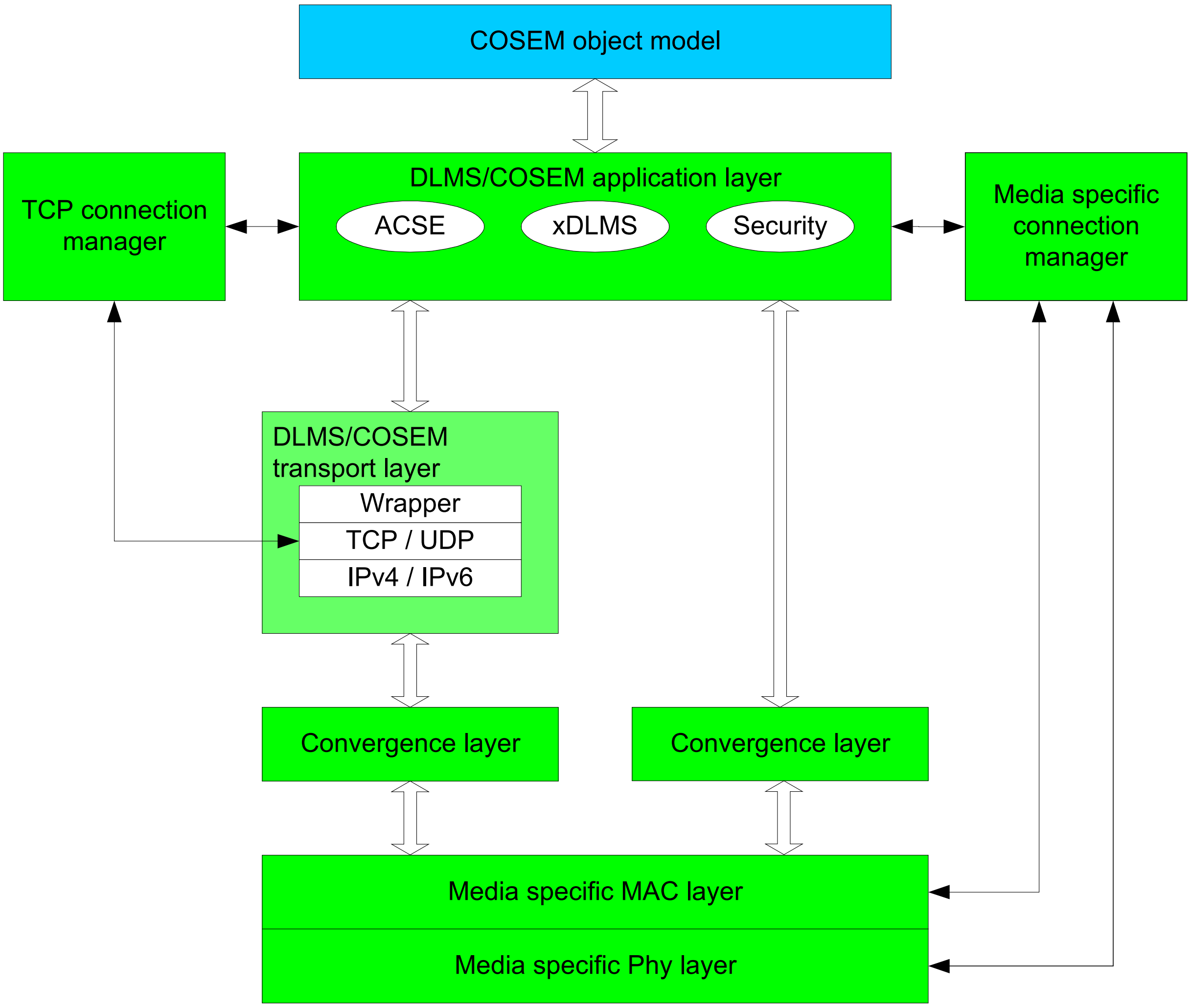


Figure 6 – DLMS/COSEM generic communication profile

Communication profiles are specified in Clause 10:

- The elements to be specified in a communication profile are specified in 10.1;
- The 3-layer, connection-oriented, HDLC based communication profile, specified in 10.2;
- The TCP-UDP/IP based communication profiles (COSEM_on_IP), specified in 10.3;
- The S-FSK PLC profile specified in the complete Green Book;
- The wired and wireless M-Bus profile, specified in the complete Green Book.

4.9 Model of a DLMS/COSEM metering system

Figure 7 shows a model of a DLMS/COSEM metering system.

Metering equipment is modelled as a set of logical devices, hosted in a single physical device. Each logical device represents a server AP and models a subset of the functionality of the metering equipment as these are seen through its communication interfaces. The various functions are modelled using COSEM objects.

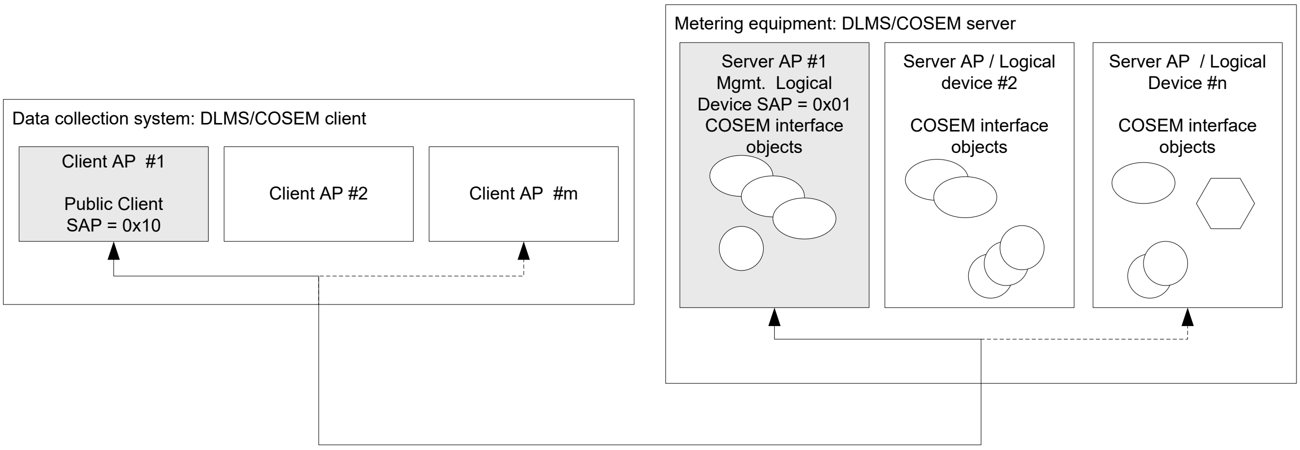


Figure 7 – Model of a DLMS/COSEM metering system

Data collection systems are modelled as a set of client Aps that may be hosted by one or several physical devices. Each client AP may have different roles and access rights, granted by the metering equipment.

The Public Client and the Management Logical Device APs have a special role and they shall be always present.

See more in DLMS UA 1000-1 Ed. 13:2019, 4.1.7 and 4.1.8.

4.10 Model of DLMS/COSEM servers

Figure 8 shows the model of two DLMS/COSEM servers as an example. One of them uses a 3-layer, CO, HDLC based communication profile, and the other one uses a TCP-UDP/IP based communication profile.

The metering equipment on the left hand side comprises “n” logical devices and supports the 3-layer, CO, HDLC based communication profile.

The DLMS/COSEM AL is supported by the HDLC based data link layer. Its main role is to provide a reliable data transfer between the peer layers. It also provides addressing of the logical devices in such a way, that each logical device is bound to a single HDLC address. The Management Logical Device is always bound to the address 0x01. To allow creating a local network so that several metering devices at a given metering site can be reached through a single access point, another address, the physical address is also provided by the data link layer. The logical device addresses are referred to as upper HDLC addresses, while the physical device address is referred to as a lower HDLC address. See also 8.4.2.

The PhL supporting the data link layer provides serial bit transmission between physical devices hosting the client and server applications. This allows using various interfaces, like RS 232, RS 485, 20 mA current loop, etc. to transfer data locally through PSTN and GSM networks etc.

The metering equipment on the right hand side comprises “m” logical devices.

The DLMS/COSEM AL is supported by the DLMS/COSEM TL, comprising the internet TCP or UDP layer and a wrapper. The main role of the wrapper is to adapt the OSI-style service set, provided by the DLMS/COSEM TL to and from TCP and UDP function calls. It also provides addressing for the logical devices, binding them to a SAP called wrapper port. The Management Logical Device is always bound to wrapper port 0x01. Finally, the wrapper provides information about the length of the APDUs transmitted, to help the peer to recognise the end of the APDU. This is necessary due the streaming nature of TCP.

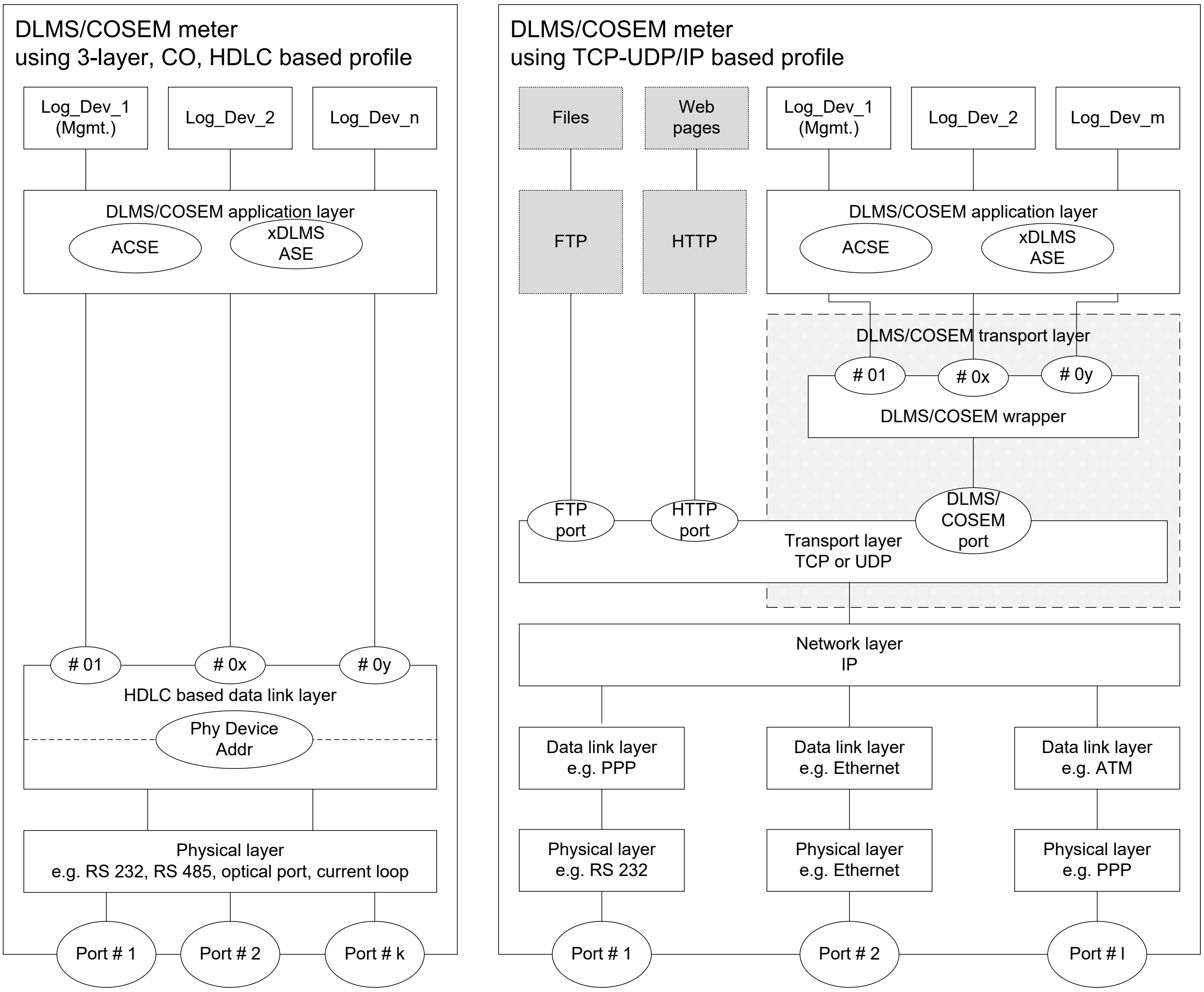


Figure 8 – DLMS/COSEM server model

Through the wrapper, the DLMS/COSEM AL is bound to a TCP or UDP port number, which is used for the DLMS/COSEM application. The presence of the TCP and UDP layers allows incorporating other internet applications, like FTP or HTTP, bound to their standard ports respectively.

The TCP layer is supported by the IP layer, which is in turn may be supported by any set of lower layers depending on the communication media to be used (for example Ethernet, PPP, IEEE 802, or IP-capable PLC lower layers, etc.).

Obviously, in a single server it is possible to implement several protocol stacks, with the common DLMS/COSEM AL being supported by distinct sets of lower layers. This allows the server to exchange data via various communication media with clients in different AAs. Such a structure would be similar to the structure of a DLMS/COSEM client show below.

4.11 Model of a DLMS/COSEM client

Figure 9 shows the model of a DLMS/COSEM client as an example.

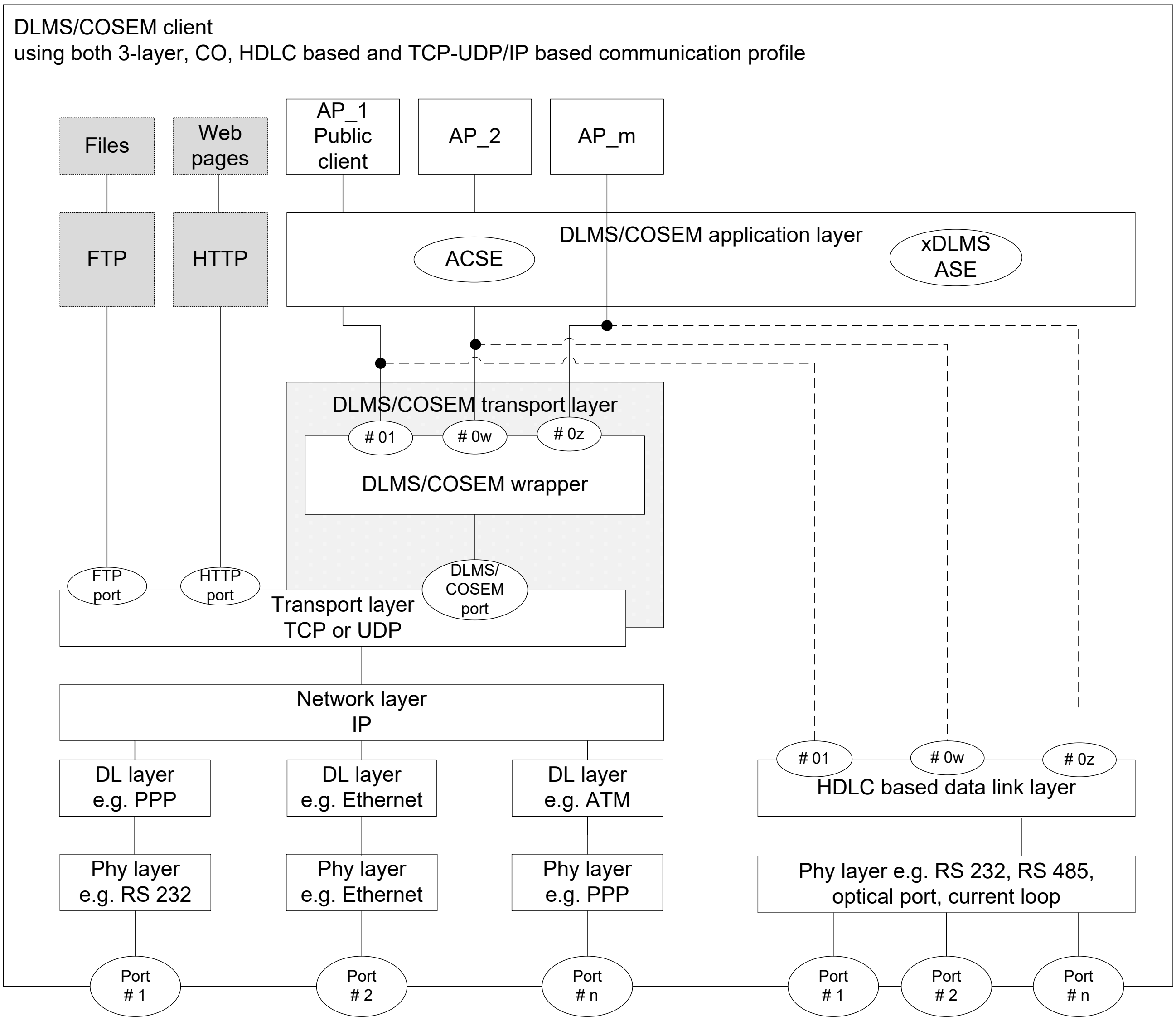


Figure 9 – Model of a DLMS/COSEM client using multiple protocol stacks

The model of the client – obviously – is very similar to the model of the servers:

- in this particular model, the DLMS/COSEM AL is supported either by the HDLC based data link layer or the DLMS/COSEM TL, meaning that the AL uses the services of one or the other as determined by the APs. In other words, the APDUs are received from or sent through the appropriate supporting layer, which in turn use the services of its supporting layer respectively;
- unlike on the server side, the addressing provided by the HDLC layer has a single level only, that of the Service Access Points (SAP) of each Application Process (AP).

As explained, client APs and server APs are identified by their SAPs. Therefore, an AA between a client and a server side AP can be identified by a pair of client and server SAPs.

The DLMS/COSEM AL may be capable to support one or more AAs simultaneously. Likewise, lower layers may be capable of supporting more than one connection with their peer layers. This allows data exchange between clients and servers simultaneously via different ports and communication media.

4.12 Interoperability and interconnectivity in DLMS/COSEM

In the DLMS/COSEM environment, interoperability and interconnectivity is defined between client and server AEs. A client and a server AE must be interoperable and interconnectable to ensure data exchange between the two systems.

Using the COSEM object model to model metering of all kinds of energy, over all communication media ensures *semantic interoperability*, i.e. an unambiguous, shared meaning between clients and servers using different communication media. The semantic elements are the COSEM objects, their logical name i.e. the OBIS code, the definition of their attributes and methods and the data types that can be used.

Using the DLMS/COSEM AL over all communication media ensures *syntactic interoperability*, which is a pre-requisite of *semantic interoperability*. Syntactic interoperability comprises the ability to establish AAs between clients and server using various application contexts, authentication mechanisms, xDLMS contexts and security contexts as well as the standard structure and encoding of all messages exchanged.

Interconnectivity is a protocol level notion: in order to be able to exchange messages, the client and the server AEs should be ***interconnectable*** and ***interconnected***.

Before the two AEs can establish an AA, they must be *interconnected*. The two AEs are interconnected, if each peer protocol layer of both sides, which needs to be connected, is connected. In order to be interconnected, the client and server AEs should be interconnectable and shall establish the required connections. Two AEs are *interconnectable* if they use the same communication profile.

With this, interconnectivity in DLMS/COSEM is ensured by the ability of the DLMS/COSEM AE to establish a connection between all peer layers, which need to be connected.

4.13 Ensuring interconnectivity: the protocol identification service

In DLMS/COSEM, AA establishment is always initiated by the client AE. However, in some cases, it may not have knowledge about the protocol stack used by an unknown server device (for example when the server has initiated the physical connection establishment). In such cases, the client AE needs to obtain information about the protocol stack implemented in the server.

A specific, application level service is available for this purpose: the protocol identification service. It is an optional application level service, allowing the client AE to obtain information – after establishing a physical connection – about the protocol stack implemented in the server. The protocol identification service, specified in the complete Green Book, uses directly the data transfer services (PH-DATA.request /.indication) of the PhL; it bypasses the other protocol layers. It is recommended to support it in all communication profiles that have access to the PhL.

4.14 System integration and meter installation

System integration is supported by DLMS/COSEM in a number of ways.

A possible process is described here.

As shown in Figure 7, the presence of a Public Client (bound to address 0x10 in any profile) is mandatory in each client system. Its main role is to reveal the structure of an unknown – for example newly installed – metering equipment. This takes place within a mandatory AA between the Public Client and the Management Logical Device, with no security precautions. Once the structure is known, data can be accessed with using the proper authentication mechanisms and cryptographic protection of the xDLMS messages and COSEM data.

When a new meter is installed in the system, it may generate an event report to the client. Once this is detected, the client can retrieve the internal structure of the meter, and then send the necessary configuration information (for example tariff schedules and installation specific parameters) to the meter. With this, the meter is ready to use.

System integration is also facilitated by the availability of the DLMS/COSEM conformance testing, described in the Yellow Book, DLMS UA 1001-1. With this, correct implementation of the specification in metering equipment can be tested and certified.

5 Physical layer services and procedures for connection-oriented asynchronous data exchange (excerpt)

NOTE The physical layer specified here is described is intended primarily for use in the 3-layer, CO, HDLC based communication profile. The physical layer of the S-FSK PLC communication profile is described in the complete Green Book. The physical layer to be used in the TCP-UDP/IP based communication profile is out of the Scope of this Technical Report.

5.1 Overview

From the external point of view, the physical layer (PhL) provides the interface between the Data Terminal Equipment, DTE, and the Data Communication Equipment, DCE, see Figure 11. Figure 10 shows a typical configuration for data exchange through a wide area network, for example the PSTN.

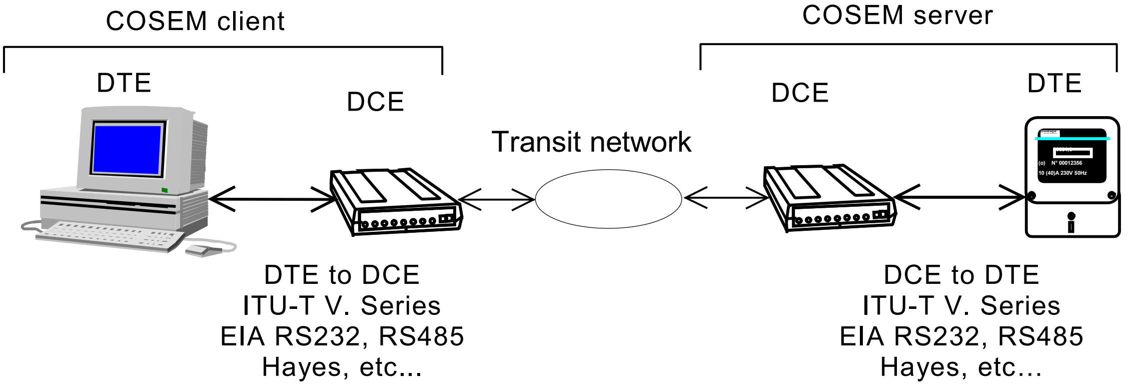


Figure 10 – Typical PSTN configuration

From the physical connection point of view, all communications involve two sets of equipment represented by the terms caller system and called system. The caller system is the system that decides to initiate a communication with a remote system known as the called system; these denominations remain valid throughout the duration of the communication. A communication is broken down into a certain number of transactions. Each transaction is represented by a transmission from the transmitter to the receiver. During the sequence of transactions, the caller and called systems take turns to act as transmitter and receiver.

From the data link point of view, the DCS normally acts as a master (primary station), taking the initiative and controlling the data flow. The metering equipment is the slave (secondary station), responding to the primary station.

From the application point of view, the DCS normally acts as a client asking for services, and the metering equipment acts as a server delivering the requested services.

The situation involving a caller client and a called server is undoubtedly the most frequent case, but a communication based on a caller server and a called client is also possible, in particular to report the occurrence of an urgent alarm.

For the purposes of local data exchange, two DTEs can be directly connected using appropriate connections. To allow using a wide variety of media, this Technical Report does not specify the PhL signals and their characteristics. However, the following assumptions are made:

- the communication is point to point or point to multipoint;
- at least half-duplex connections are possible;
- asynchronous transmission with 1 start bit, 8 data bits, no parity and 1 stop bit (8N1).

From the internal point of view, the PhL is the lowest layer in the protocol stack.

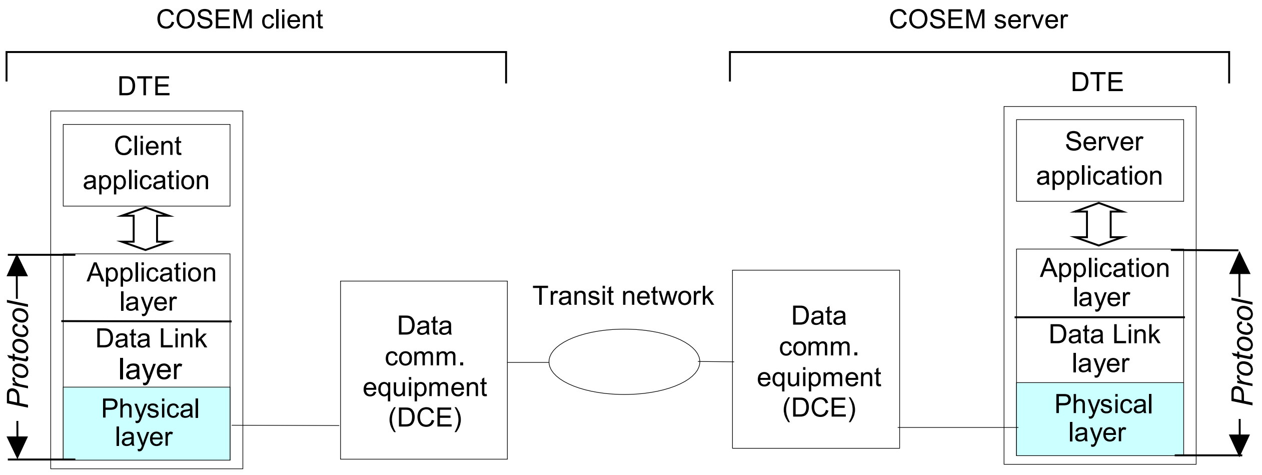


Figure 11 – The location of the physical layer

In the following, the services of the PhL towards its peer layer(s) and the upper layers, as well as the protocol of the PhL are defined.

5.2 Service specification

5.2.1 List of services

ITU-T X.211 defines a set of capabilities to be made available by the PhL over the physical media. These capabilities are available via services, as follows:

• Connection establishment/release related services:	PH-CONNECT, PH-ABORT;
• Data transfer services:	PH-DATA;
• Layer management services.	

Layer management services are used by or provided for the layer management process, which is part of the AP. Some examples are given below:

- PH-INITIALIZE.request / PH-INITIALIZE.confirm;
- PH-GET_VALUE.request / PH-GET_VALUE.confirm;
- PH-SET_VALUE.request / PH-SET_VALUE.confirm;
- PH-LM_EVENT.indication.

As these services are of local importance only, their definition is not within the Scope of this Technical Report.

5.2.2 Use of the physical layer services

Figure 12 shows how different service users use the service primitives of the PhL. As it can be seen, the physical connection establishment/release services are used by and provided for the physical connection manager AP, and not the data link layer. The reasons for this are explained in the complete Green Book.

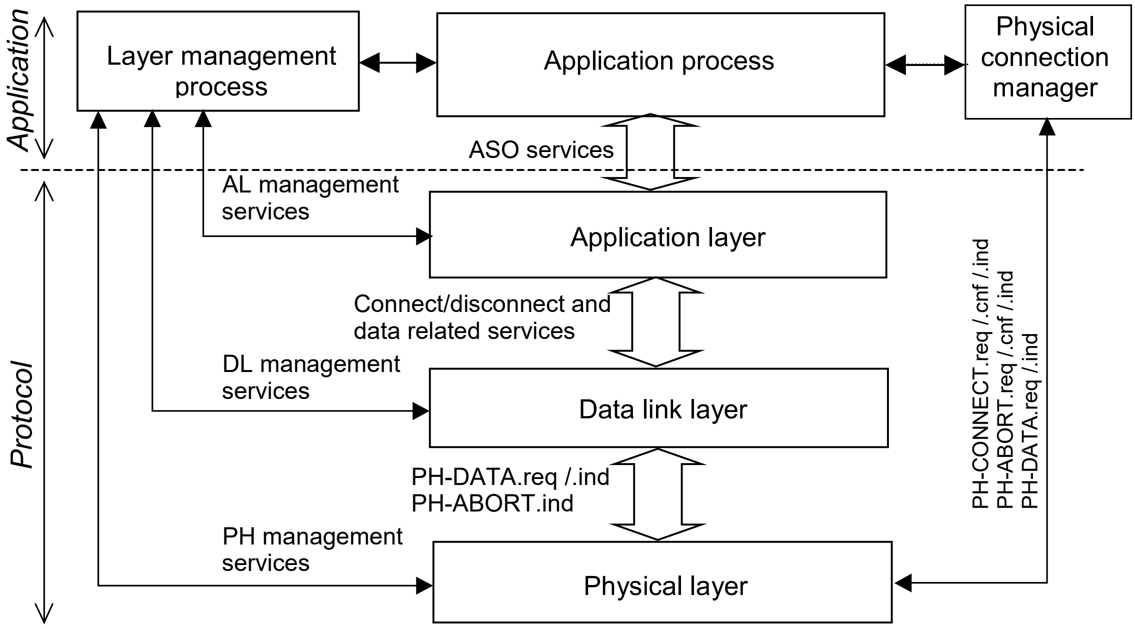


Figure 12 – Protocol layer services of the COSEM 3-layer connection-oriented profile

For more information see the complete Green Book.

6 Direct Local Connection (excerpt)

6.1 Introduction

This Clause 6 is an excerpt of IEC 62056-21 describing hardware and protocol specifications for local meter data exchange. In such systems, a hand-held unit (HHU) or a unit with equivalent functions is connected to a tariff device or a group of devices. Only COSEM related items are described here. The complete information can be found in IEC 62056-21.

NOTE Support for local interface based on IEC 62056-21 is not mandated within DLMS/COSEM. Local connection using HDLC *ab initio*, or PPP, or no local interface, are equally acceptable.

6.2 METERING HDLC protocol using protocol mode E for direct local data exchange

The protocol stack as described in Clauses 5, 8 and 9 of this Technical Report shall be used.

The switch to the baud rate Z shall be at the same place as for protocol mode C. The switch confirm message, which has the same structure as the acknowledgement/option select message, is therefore at the new baud rate but still with parity (7E1). After the acknowledgement, the binary mode (8N1) will be established.

As the server acknowledgement string is a constant in the server's program, it could be easily possible to switch to the baud rate and the binary mode (Z Bd. 8N1) at the same time. The characters ACK 2 Z 2 CR LF in that case shall be replaced by their 8 bit equivalents by adding the correct parity bit in order to simulate their 7E1 equivalents. This alternative method is not visible to the client; both have an equivalent behaviour (see also the complete Green Book).

A client, which is not able to support protocol HDLC mode E (W=2) will answer in a protocol mode as defined by Y (normally protocol mode C).

The enhanced capability of the server (tariff device) is communicated with the escape sequence "\W" which is part of the meter identification string (see items 14), 23) and 24) in IEC 62056-21:2002, 6.3.14 [2].

[2] W = @ is used for country specific applications

6.3 Overview

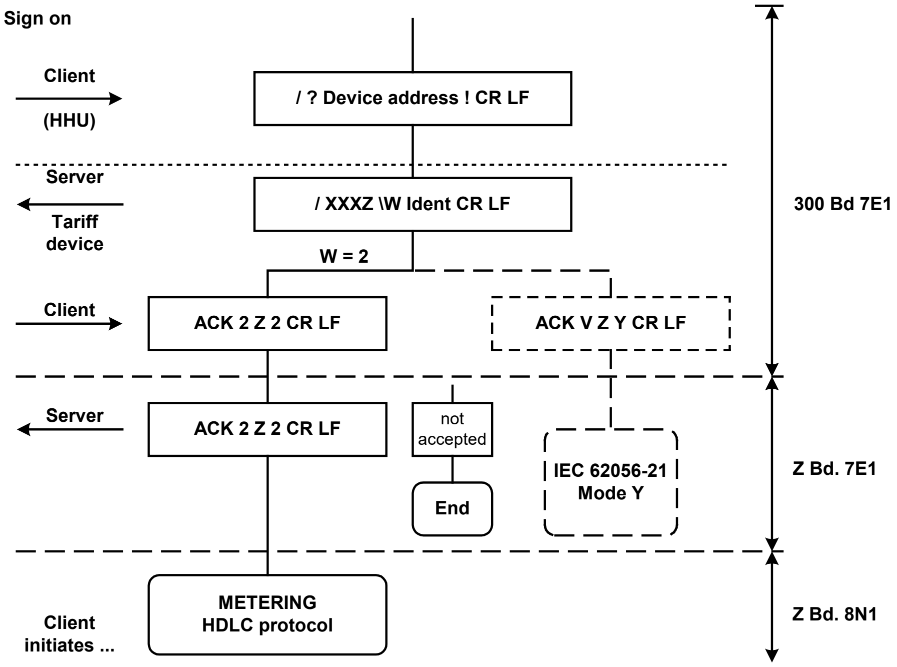


Figure 13 – Entering protocol mode E (HDLC)

For more information see the complete Green Book.

7 DLMS/COSEM transport layer for IP networks

7.1 Scope

This Clause 7 specifies a connection-less and a connection oriented transport layer (TL) for DLMS/COSEM communication profiles used on IP networks.

These TLs provide OSI-style services to the service user DLMS/COSEM AL. The connection-less TL is based on the Internet Standard User Datagram Protocol (UDP). The connection-oriented TL is based on the Internet Standard Transmission Control Protocol (TCP).

The DLMS/COSEM TL consists of the UDP or TCP transport layer TCP and an additional sublayer, called wrapper.

Subclause 7.5 shows how the OSI-style TL services can be converted to and from UDP and TCP function calls.

7.2 Overview

In the DLMS/COSEM_on_IP profiles, the DLMS/COSEM AL uses the services of one of these TLs, which use then the services of the Internet Protocol (IP) network layer to communicate with other nodes connected to the IP network.

When used in these profiles, the DLMS/COSEM AL can be considered as another Internet standard application protocol (like the well-known HTTP, FTP or SNMP) and it may co-exist with other Internet application protocols, as it is shown in Figure 14.

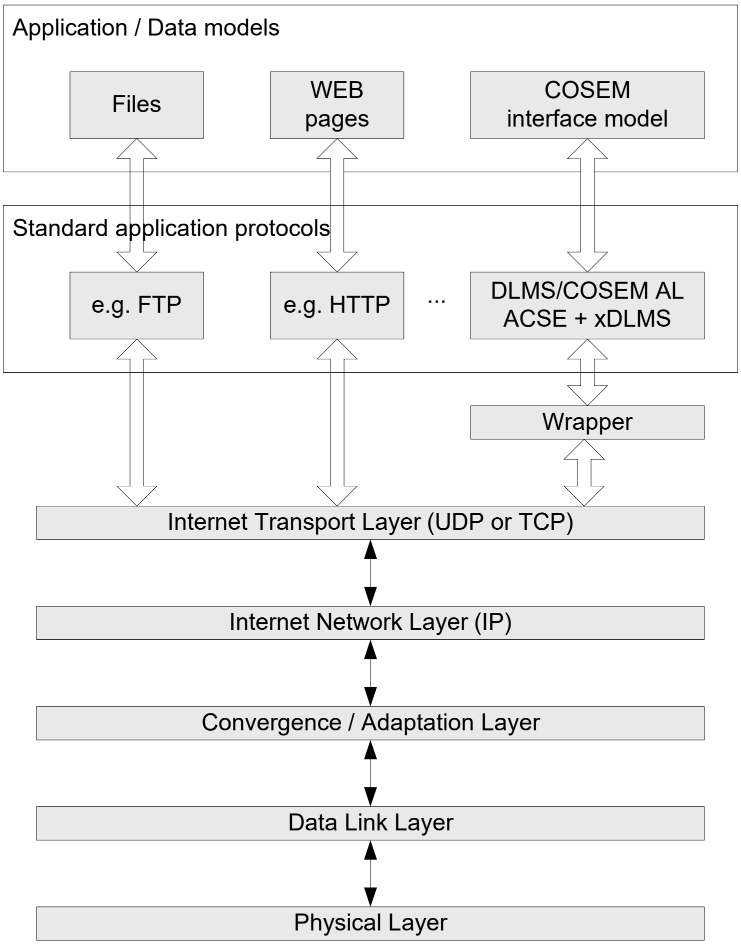

Application / Data models
Files
WEB pages
COSEM interface model
Standard application protocols
e.g. FTP
e.g. HTTP
...
DLMS/COSEM AL ACSE + xDLMS
Wrapper
Internet Transport Layer (UDP or TCP)
Internet Network Layer (IP)
Convergence / Adaptation Layer
Data Link Layer
Physical Layer

Figure 14 – DLMS/COSEM as a standard Internet application protocol

For DLMS/COSEM, the following port numbers have been registered by the IANA. See http://www.iana.org/assignments/port-numbers:

dlms/cosem	4059/TCP	DLMS/COSEM;
dlms/cosem	4059/UDP	DLMS/COSEM.

As the DLMS/COSEM AL specified in Clause 9 uses and provides OSI-style services, a wrapper has been introduced between the UDP/TCP layers and the DLMS/COSEM AL. Therefore, the DLMS/COSEM TLs consist of a wrapper sublayer and the UDP or TCP TL. The wrapper sublayer is a lightweight, nearly state-less entity: its main function is to adapt the OSI-style service set, provided by the DLMS/COSEM TL, to UDP or TCP function calls and vice versa.

In addition, the wrapper sublayer has the following functions:

- it provides an additional addressing capability (wPort) on top of the UDP/TCP port;
- it provides information about the length of the data transported. This feature helps the sender to send and the receiver to recognize the reception of a complete APDU, which may be sent and received in multiple TCP packets.

As specified in the complete Green Book, the DLMS/COSEM AL is listening only on one UDP or TCP port. On the other hand, as shown in 4.9 and in DLMS UA 1000-1, a physical device may host several client or server APs. The additional addressing capability provided by the wrapper sublayer allows addressing these APs.

The structure of the DLMS/COSEM TL and their place in DLMS/COSEM_on_IP is shown in Figure 15.

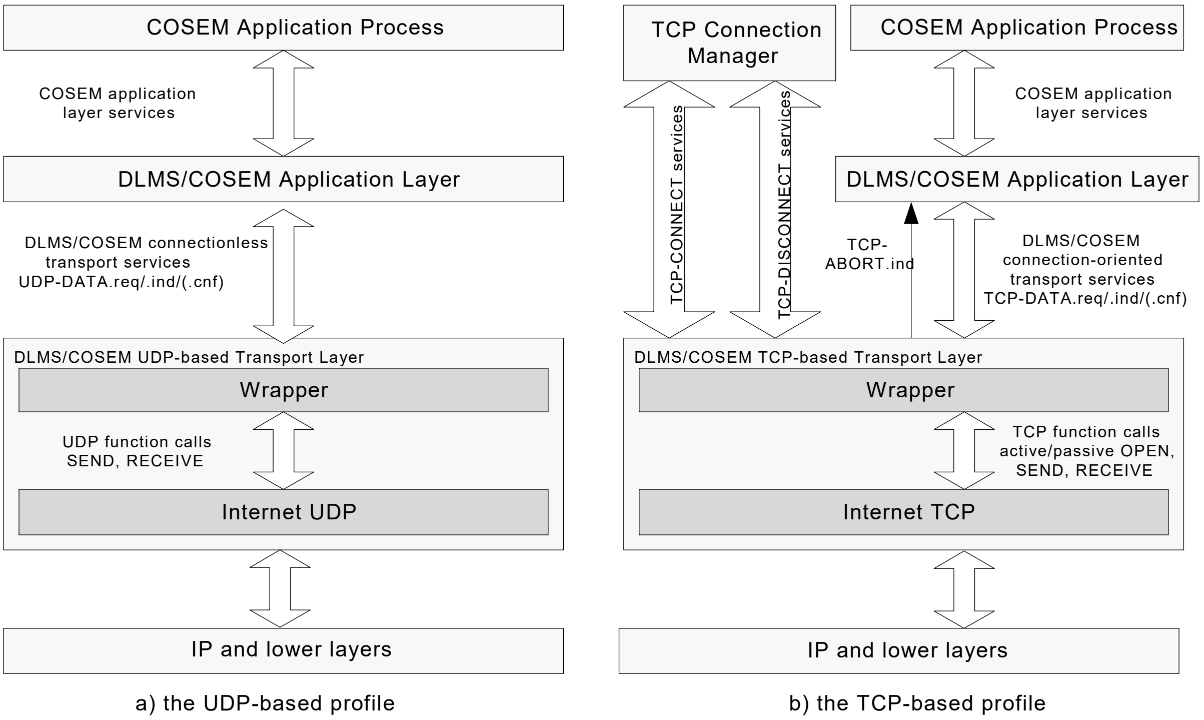

COSEM Application Process
COSEM application layer services
DLMS/COSEM Application Layer
DLMS/COSEM connectionless transport services UDP-DATA.req/.ind/(.cnf)
DLMS/COSEM UDP-based Transport Layer
Wrapper
UDP function calls SEND, RECEIVE
Internet UDP
IP and lower layers
a) the UDP-based profile
TCP Connection Manager
COSEM Application Process
TCP-CONNECT services
TCP-DISCONNECT services
COSEM application layer services
DLMS/COSEM Application Layer
TCP-ABORT.ind
DLMS/COSEM connection-oriented transport services TCP-DATA.req/.ind/(.cnf)
DLMS/COSEM TCP-based Transport Layer
Wrapper
TCP function calls active/passive OPEN, SEND, RECEIVE
Internet TCP
IP and lower layers
b) the TCP-based profile

Figure 15 – Transport layers of the DLMS/COSEM_on_IP profile

The service user of both the UDP-DATA and the TCP-DATA services is the DLMS/COSEM AL. On the other hand, the service user of the TCP-CONNECT and TCP-DISCONNECT services is the TCP Connection Manager Process. The DLMS/COSEM TCP-based TL also provides a TCP-ABORT service to the service user DLMS/COSEM AL.

7.3 The DLMS/COSEM connection-less, UDP-based transport layer

7.3.1 General

The DLMS/COSEM connection-less TL is based on the User Datagram Protocol (UDP) as specified in STD0006.

UDP provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. On the one hand, the protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. On the other hand, UDP is simple, it adds a minimum of overhead, and it is efficient and easy to use. Several well-known Internet applications, like SNMP, DHCP, TFTP, etc. take advantage of these performance benefits, either because some datagram applications do not need to be reliable or because the required reliability mechanism is ensured by the application itself. Request/response type applications, like a confirmed COSEM application association established on the DLMS/COSEM UDP-based TL, then invoking confirmed xDLMS data transfer services is a good example for this second category. Another advantage of UDP is that being connection-less, it is easily capable of multi- and broadcasting.

UDP basically provides an upper interface to the IP layer, with an additional identification capability, the UDP port number. This allows distinguishing between APs, hosted in the same physical device and identified by its IP address.

For more information see the complete Green Book.

7.4 The DLMS/COSEM connection-oriented, TCP-based transport layer

7.4.1 General

The DLMS/COSEM connection-oriented TL is based on the connection-oriented Internet transport protocol, called Transmission Control Protocol. TCP is an end-to-end reliable protocol. This reliability is ensured by a conceptual “virtual circuit”, using a method called PAR, Positive Acknowledgement with Retransmission. It provides acknowledged data delivery, error detection and data re-transmission after an acknowledgement time-out, etc. Therefore it deals with lost, delayed, duplicated or erroneous data packets. In addition, TCP offers an efficient flow control mechanism and full-duplex operation, too.

TCP, as a connection-oriented transfer protocol involves three phases: connection establishment, data exchange and connection release. Consequently, the DLMS/COSEM TCP-based TL provides OSI-style services to the service user(s) for all three phases:

- for the connection establishment phase, the TCP-CONNECT service is provided to the service user TCP connection manager process;
- for the data transfer phase, the TCP-DATA service is provided to the service user DLMS/COSEM AL;
- for the connection closing phase, the TCP-DISCONNECT service is provided to the service user TCP connection manager process;
- in addition, a TCP-ABORT service is provided to the service user DLMS/COSEM AL.

The DLMS/COSEM connection-oriented, TCP-based TL contains the same wrapper sublayer as the DLMS/COSEM UDP-based TL. In addition to transforming OSI-style services to and from TCP function calls, this wrapper provides additional addressing and length information.

The DLMS/COSEM connection-oriented, TCP-based TL is specified in terms of services and protocols. The conversion between OSI-style services and TCP function calls is presented in 7.5.

For more information see the complete Green Book.

7.5 Converting OSI-style TL services to and from RFC-style TCP function calls

7.5.1 Transport layer and TCP connection establishment

As specified in STD0007, a TCP connection is established by calling the OPEN function. This function can be called in *active* or *passive* manner.

According to the TCP connection state diagram (Figure 16) a *passive* OPEN takes the caller device to the LISTEN state, waiting for a connection request from any remote TCP and port.

An *active* OPEN call makes the TCP to establish the connection to a remote TCP.

The establishment of a TCP Connection is performed by using the so-called “Three-way handshake” procedure. This is initiated by one TCP calling an *active* OPEN and responded by another TCP, the one, which has already been called a *passive* OPEN and consequently is in the LISTEN state.

The message sequence – and the state transitions corresponding to that message exchange – for this “three-way handshake” procedure are shown in Figure 17.

This process, consisting of three messages, establishes the TCP connection and “synchronizes” the initial sequence numbers at both sides. This mechanism has been carefully designed to guarantee, that both sides are ready to transmit data and know that the other side is ready to transmit as well. Note that the procedure also works if two TCPs simultaneously initiate the procedure.

NOTE Sequence numbers are part of the TCP packet, and are fundamental to reliable data transfer. For more details about sequence numbers (or other TCP related issues), please refer to STD0007.

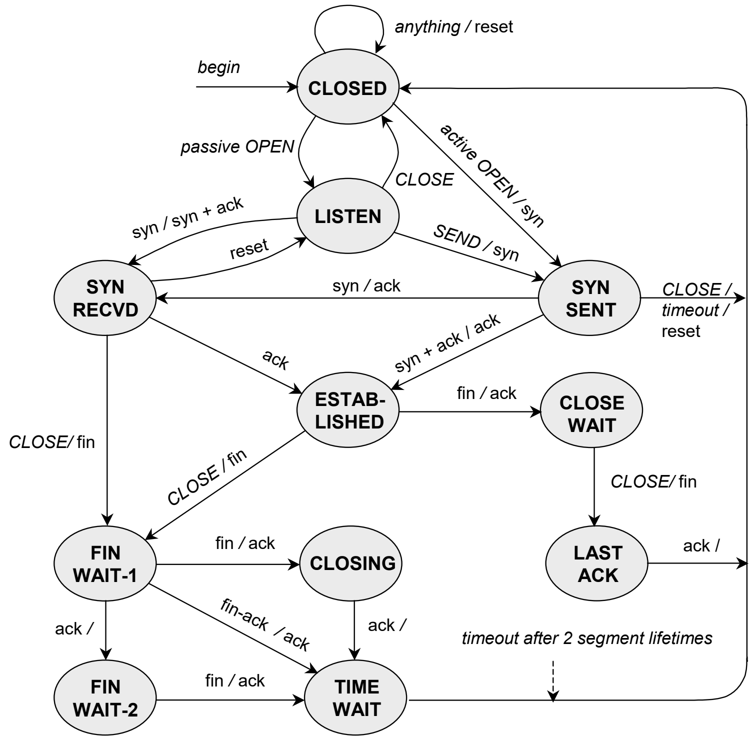
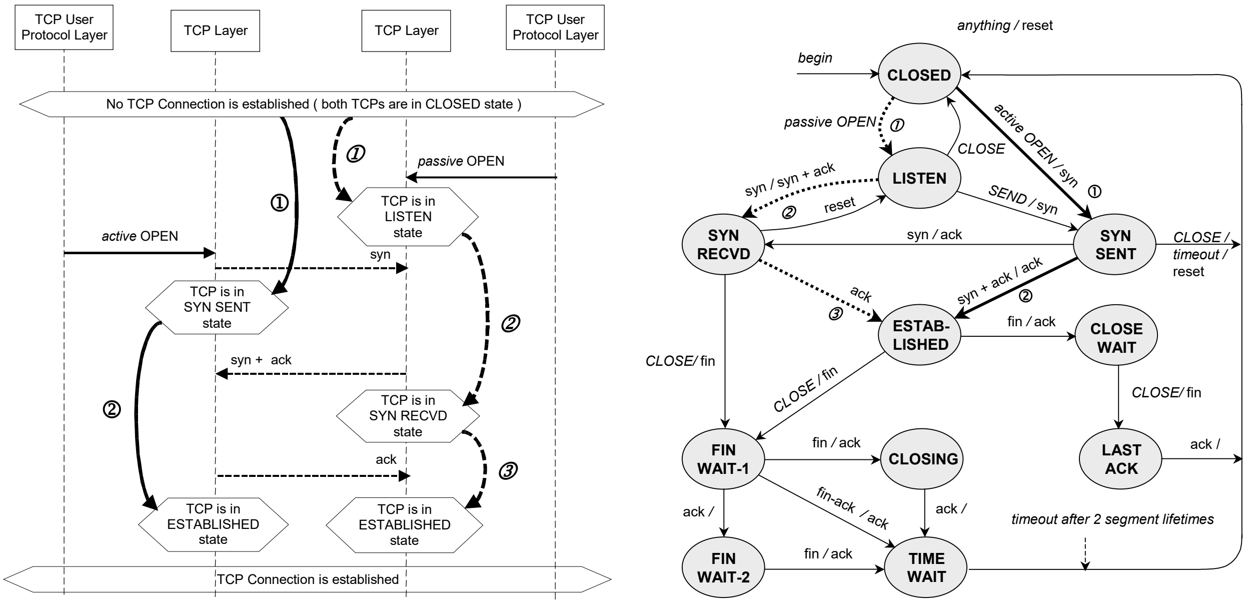


Figure 16 – TCP connection state diagram



NOTE In the case of the DLMS/COSEM transport layer, the TCP user protocol layer is the wrapper sublayer.

Figure 17 – MSC and state transitions for establishing a transport layer and TCP connection

For more information see the complete Green Book.

8 Data Link Layer using the HDLC protocol

8.1 Overview

8.1.1 General

This chapter specifies the data link layer for the 3-layer, connection-oriented, HDLC based, asynchronous communication profile.

This specification supports the following communication environments:

- point-to-point and point-to-multipoint configurations;
- dedicated and switched data transmission facilities;
- half-duplex and full-duplex connections;
- asynchronous start/stop transmission, with 1 start bit, 8 data bits, no parity, 1 stop bit.

Two special procedures are also defined:

- transferring of separately received Service User layer PDU parts from the server to the client in a transparent manner. The server side Service user layer can give its PDU to the data link layer in fragments and the data link layer can hide this fragmentation from the client, see the complete Green Book:
- event reporting, by sending UI frames from the secondary station to the primary station, see the complete Green Book.

Clause 4 gives an explanation of the role of data models and protocols in meter data exchange.

8.1.2 Structure of the data link layer

In order to ensure a coherent data link layer service specification for both connection-oriented and connectionless operation modes, the data link layer is divided into two sublayers: the Logical Link Control (LLC) sublayer and the Medium Access Control (MAC) sublayer.

The LLC sublayer is based on ISO/IEC 8802-2.

The presence of this sublayer in the connection-oriented profile is somewhat artificial: it is used as a kind of protocol selector, and the ‘real’ data link layer connection is ensured by the MAC sublayer. It can be considered that the standard LLC sublayer is used in an extended class I operation, where the LLC sublayer provides the standard data link connectionless services to its service user layer via a connection-oriented MAC sublayer, which executes the services.

The MAC sublayer – the major part of this data link layer specification – is based on ISO/IEC 13239. The second edition of that standard includes a number of enhancements compared to the original HDLC standard, for example in the areas of addressing, error protection, and segmentation. The third edition incorporates a new frame format, which meets the requirements of the environment found in telemetry applications for electricity metering and similar industries.

For the purpose of this Technical Report, the following selections from the HDLC standard have been made:

- unbalanced connection-mode data link operation;

NOTE In the DLMS/COSEM environment, the choice of an unbalanced mode of operation is natural: it is the consequence of the fact that communication in this environment is based on the client/server model.

- two-way alternate data transfer , TWA;
- the selected HDLC class of procedures is UNC – Unbalanced operation Normal response mode Class – extended with UI frames;
- frame format type 3;
- non-basic frame format transparency.

In the unbalanced connection-mode data link operation two or more stations are involved. The primary station assumes responsibility for the organization of data flow and for unrecoverable data link level error conditions, by sending command and supervisory frames. The secondary station(s) respond(s) by sending response frames.

NOTE In the context of DLMS/COSEM the primary station is often, but does not have to be, the client.

The basic repertoire of commands and responses of the UNC class of procedures is extended with the UI frame to support multicasting and broadcasting and non-solicited information transfer from server to the client.

Using the unbalanced connection-mode data link operation implies that the client and server side data link layers are different in terms of the sets of HDLC frames and their state machines.

8.1.3 Specification method

Sublayers of the data link layer are specified in terms of ***services*** and ***protocols***.

Service specifications cover the services required of, or by, the given sublayer at the logical interfaces with the neighbouring other sublayer or layer, using connection-oriented procedures. Services are the standard way to specify communications between protocol layers. Through the use of four types of transactions, commonly known as service primitives (Request, Indication, Response and Confirm) the service provider co-ordinates and manages the communication between the users. Using service primitives is an abstract, implementation-independent way to specify the transactions between protocol layers. Given this abstract nature of the primitives, their use makes good sense for the following reasons:

- they permit a common convention to be used between layers, without regard to specific operating systems and specific languages;
- they give the implementers a choice of how to implement the service primitives on a specific machine.

Service primitives include service parameters. There are three classes of service parameters:

- parameters transmitted to the peer layer, becoming part of the transmitted frame, for example addresses, control information;
- parameters, which have only local significance;
- parameters, which are transmitted transparently across the data link layer to the user of the data link.

NOTE Data link layer management services are explained in the complete Green Book.

This Technical Report specifies values for parameters of the first category only.

As the services of the data link layer – called DL services – are in fact provided by the MAC sublayer i.e. the MA services, the two service sets are specified together in 8.2 for a concise presentation.

Protocol specifications for a protocol layer / sublayer include:

- the specification of the procedures for the transmission of the set of messages exchanged between peer layers;
- the procedures for the correct interpretation of protocol control information;
- the layer behaviour.

Protocol specifications for a protocol layer / sublayer do not include:

- the structure and the meaning of the information which is transmitted by means of the layer (Information field, User data subfield);
- the identity of the Service User layer;
- the manner in which the Service User layer operation is accomplished as a result of exchanging Data Link messages;
- the interactions that are the result of using the protocol layer.

The protocol for the LLC sublayer is specified in 8.3 and the protocol for the MAC sublayer is specified in 8.4.

As the MAC sublayer behaviour is quite complex, some aspects of the service invocation handling are discussed in the service specification part, although these are normally part of the protocol specification.

8.2 Service specification

8.2.1 General

This clause specifies the services required of the data link layer by the service user layer, using connection-oriented procedures.

All DL services are, in fact, provided by the MAC sublayer: the LLC sublayer transparently transmits the DL-CONNECT.xxx service primitives to/from the “real” service provider MAC sublayer as the appropriate MA-CONNECT.xxx service primitive.

As the client and the server side LLC and MAC sublayers are different, service primitives are specified for both sides.

The addressing scheme for the MAC sublayer is specified in 8.4.2.

For more information see the complete Green Book.

8.3 Protocol specification for the LLC sublayer

8.3.1 Role of the LLC sublayer

The LLC sublayer transmits LSDUs transparently between its service user layer and the MAC sublayer.

8.3.2 LLC PDU format

The standard LLC PDU format is shown in Figure 18:

Destination (remote) LSAP	Source (local) LSAP	Control	Information
8 bits	8 bits	8 or 16 bits	n*8 bits

Figure 18 – The ISO/IEC 8802-2 LLC PDU format

For the purposes of DLMS/COSEM, this LLC PDU format is used as shown on Figure 19:

Destination (remote) LSAP	Source (local) LSAP	LLC_Quality	Information
8 bits: 0xE6	8 bits: 0xE6 or 0xE7	8 bits: 0x00	n*8 bits

Figure 19 – LLC format as used in DLMS/COSEM

- The value of the Destination_LSAP is 0xE6;
- The value of the Source_LSAP is 0xE6 or 0xE7. The least significant bit is used as a command/response identifier. When set to 0, it identifies a ‘command’ and when set to 1 it identifies a “response”;
- The Control byte is referred here to as the LLC_Quality parameter. It is reserved for future use. Its value is administered by the DLMS UA. Currently, it must be set always to 0x00;
- The information field consists of an integral number (including zero) of octets and it carries the LSDU.

The destination LSAP 0xFF is used for broadcasting purposes. Devices in this environment shall never send messages with this broadcast address, but they shall accept messages containing this broadcast destination address as if it would be addressed to them.

For more information see the complete Green Book.

8.4 Protocol specification for the MAC sublayer

8.4.1 The MAC PDU and the HDLC frame

8.4.1.1 HDLC frame format type 3

The MAC sublayer uses the HDLC frame format type 3 as defined in Annex H.4 of ISO/IEC 13239. It is shown on Figure 20:

Flag	Frame format	Dest. address	Src. address	Control	HCS	Information	FCS	Flag

Figure 20 – MAC sublayer frame format (HDLC frame format type 3)

This frame format is used in those environments where additional error protection, identification of both the source and the destination, and/or longer frame sizes are needed. Type 3 requires the use of the segmentation subfield, thus reducing the length field to 11 bits. Frames that do not have an information field, for example as with some supervisory frames, or an information field of zero length do not contain an HCS and an FCS, only an FCS. The HCS and FCS polynomials will be the same. The HCS shall be 2 octets in length.

The elements of the frame are described in the following clauses.

8.4.1.2 Flag field

The length of the flag field is one byte and its value is 0x7E. When two or more frames are transmitted continuously, a single flag is used as both the closing flag of one frame and the opening flag of the next frame, as it is shown in Figure 21.

NOTE Frames are transmitted continuously when the period of time between two transmitted characters does not exceed the specified max. inter-octet time. See the complete Green Book.

Flag	Frame I	Flag	Frame I+1	Flag	Frame I+2		Flag

Figure 21 – Multiple frames

8.4.1.3 Frame format field

The length of the frame format field is two bytes. It consists of three sub-fields referred to as the Format type sub-field (4 bit), the Segmentation bit (S, 1 bit) and the frame length sub-field (11 bit), as it is shown in Figure 22:

MSB LSB

1	0	1	0	S	L	L	L	L	L	L	L	L	L	L	L

Format type Frame length sub-field

Figure 22 – The frame format field

The value of the format type sub-field is 1010 (binary), which identifies a frame format type 3 as defined in 8.4.1.1.

Rules of using the segmentation bit are defined in the complete Green Book.

The value of the frame length subfield is the count of octets in the frame excluding the opening and closing frame flag sequences.

8.4.1.4 Destination and source address fields

There are exactly two address fields in this frame: a destination and a source address field.

8.4.1.5 Control field

The length of the control field is one byte. It indicates the type of commands or responses, and contains sequence numbers, where appropriate (frames I, RR and RNR).

8.4.1.6 Header check sequence (HCS) field

The length of the HCS field is two bytes. This check sequence is applied to only the header, i.e., the bits between the opening flag sequence and the header check sequence. Frames that do not have an information field or have an empty information field, e.g., as with some supervisory frames, do not contain an HCS and FCS, only an FCS. The HCS is calculated in the same way as the FCS; see 8.5.

8.4.1.7 Information field

The information field may be any sequence of bytes. In the case of data frames (I and UI frames), it carries the MSDU.

8.4.1.8 Frame check sequence (FCS) field

The length of the FCS field is two bytes. Unless otherwise noted, the frame checking sequence is calculated for the entire length of the frame, excluding the opening flag, the FCS and any start and stop elements (start/stop transmission). Guidelines to calculate the FCS are given in 8.5.

8.4.2 MAC addressing

8.4.2.1 Use of extended addressing

As specified in ISO/IEC 13239:2002, 4.7.1, The address field range can be extended by reserving the first transmitted bit (low-order) of each address octet which would then be set to binary zero to indicate that the following octet is an extension of the address field. The format of the extended octet(s) shall be the same as that of the first octet. Thus, the address field may be recursively extended. The last octet of an address field is indicted by setting the low-order bit to binary one.

When extension is used, the presence of a binary "1" in the first transmitted bit of the first address octet indicates that only one address octet is being used. The use of address extension thus restricts the range of single octet addresses to 0x7F and for two octet addresses to 0...0x3FFF.

8.4.2.2 Address field structure

The HDLC frame format type 3 (see 8.4.1.1) contains two address fields: a destination and a source HDLC address. Depending on the direction of the data transfer, both the client and the server addresses can be destination or source addresses.

The client address shall always be expressed on one byte.

The server address – to enable addressing more than one logical device within a single physical device and to support the multi-drop configuration – may be divided into two parts:

- the upper HDLC address is used to address a Logical Device (a separately addressable entity within a physical device);
- the lower HDLC address is used to address a Physical Device (a physical device on the multi-drop).

The upper HDLC address shall always be present. The lower HDLC address may be omitted if it is not required.

The HDLC address extension mechanism applies to both parts. This mechanism specifies variable length address fields, but for the purpose of this protocol, the length of a complete server address field is restricted to be one, two or four bytes long, as shown on Figure 23. The server may support more than one addressing scheme. Individual, multicast and broadcast addressing facilities are provided for both the upper and the lower HDLC address.

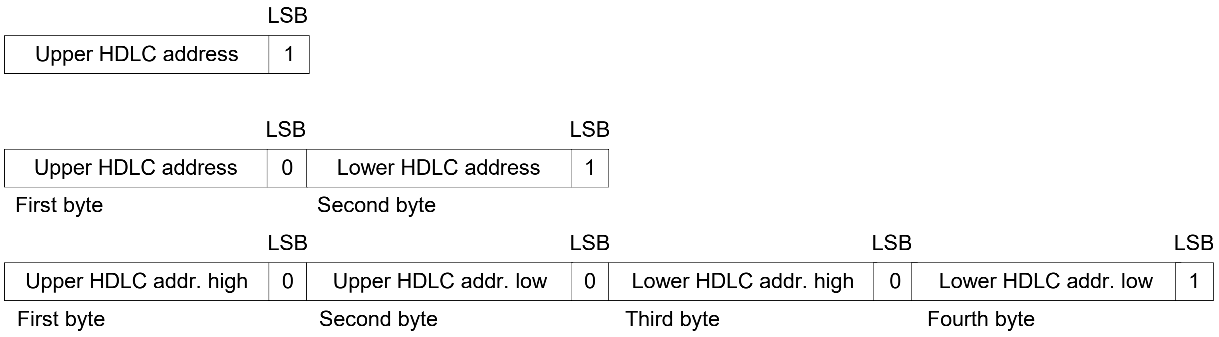


Figure 23 – Valid server address structures

8.4.2.3 Reserved special HDLC addresses

The following special HDLC addresses are reserved:

Table 2 – Table of reserved client addresses

Reserved HDLC addresses	
No-station	0x00
Client Management Process	0x01
Public Client	0x10
Open for client SAP assignment	0x02 …0x0F
	0x11… 0xFF

Table 3 – Table of reserved server addresses

Reserved upper HDLC addresses		
	One byte address	Two byte address
No-station	0x00	0x0000
Management Logical Device	0x01	0x0001
Reserved for future use	0x02…0x0F	0x0002..0x000F
Open for server SAP assignment	0x10…0x7E	0x0010…0x3FFE
All-station (Broadcast)	0x7F	0x3FFF
Reserved lower HDLC addresses		
No-station	0x00	0x0000
Reserved for future use	0x01…0x0F	0x0001…0x000F
Open for server SAP assignment	0x10…0x7D	0x0010…0x3FFD
CALLING [a]) Physical Device	0x7E	0x3FFE
All-station (Broadcast)	0x7F	0x3FFF
[a] The meaning of the CALLING Physical Device is discussed in 8.4.2.4.		

In the table above, the effect of the address extension bits is not taken into account. Their use is illustrated with the following example:

Client HDLC Address = 0x3A = 00111010_B

Server HDLC Address (using four bytes addressing)

lower HDLC Address = 0x3FFF = 0011111111111111_B All-station (Broadcast) Address

upper HDLC Address = 0x1234 = 0001001000110100_B

The address fields of the message shall contain the following octets:

Server address								Client address	
Upper HDLC high		Upper HDLC low		Lower HDLC high		Lower HDLC low		HDLC address	
LSB		LSB		LSB		LSB		LSB	
0 1 0 0 1 0 0	0	0 1 1 0 1 0 0	0	1 1 1 1 1 1 1	0	1 1 1 1 1 1 1	1	0 1 1 1 0 1 0	1
First byte		Second byte		Third byte		Fourth byte		Fifth byte	
Destination address								Source address	

Figure 24 – Address example

8.4.2.4 Handling special addresses

The following MAC address types and specific MAC addresses are specified:

- individual addresses;
- group addresses;
- the All-station address;
- the No-station address;
- the CALLING Physical Device address;
- the Management Logical Device Address (the presence of this logical device is mandatory).

The following rules apply:

- group Address management is not within the Scope of this Technical Report;
- the Source Address field of a valid HDLC frame may not contain either the All-station or the No-station address. If an HDLC frame is received with it, it shall be considered as an invalid frame;
- only HDLC frames transmitted from the primary station towards the secondary station(s) may contain the All-station or the No-station in the Destination Address field;
- broadcast and multicast I frames shall be discarded;
- the P/F bit of messages with All-station, No-station or Group address in the Destination Address field shall be set to FALSE. UI frames containing an All-station, No-station or Group address with P == TRUE shall be discarded;
- the CALLING Physical Device address is a special address to support event reporting; see the complete Green Book. It is reserved to reference the server station initiating a physical connection to the client station. It is not the station's own physical address; therefore no station shall be configured to have the CALLING Physical Address as its own physical address.

8.4.2.5 Handling inopportune address lengths in the server

Frames received by the server may contain addresses with a different length than what is expected. In such cases, the following rules apply:

- as client addresses are specified to be one byte, frames that contain more than one byte in the source address field shall be discarded;
- destination addresses (DA) shall be handled according to Table 4.

Table 4 – Handling inopportune address lengths

Length of the DA field received	Length of the DA field expected	Behaviour
1 byte	2 bytes	The frame shall be discarded.
1 byte	4 bytes	The frame shall be discarded.
2 bytes	1 byte	The frame is not discarded only if the lower MAC Address is equal to the All-station address. In this case, it shall be given to the Logical Device(s) designated by the upper MAC Address field.
2 bytes	4 bytes	The value of the one-byte lower and upper MAC addresses received shall be converted into a two + two byte address. The frame shall be taken into account as if it was received using a 4-byte DA field.
4 bytes	1 byte	The frame is not discarded only if the both the lower and upper MAC Addresses are equal to the All-station address.
4 bytes	2 bytes	If the lower MAC Address is equal to the All-station address, the frame shall be accepted only if the upper MAC Address is also equal to the All-station address. If the lower MAC address is equal to the CALLING Physical Device address, the frame shall be accepted only if the upper MAC Address is equal to the Management Logical Device Address and the CALLING DEVICE layer parameter – see the complete Green Book – is set to TRUE. In any other case, the frame received shall be discarded.
3 or more than 4 bytes	N.A.	The frame shall be discarded.

8.4.3 Command and response frames

8.4.3.1 Selected repertoire and control field format

This Technical Report uses the UNC basic repertoire of commands and responses, extended with the UI commands and responses, as defined in ISO/IEC 13239. The encoding of the command/response control fields shall be modulo 8, as specified in 5.5 of ISO/IEC 13239), shown in Table 5.

Table 5 – Control field bit assignments of command and response frames

		MSB LSB
Command	**Response**	
I	I	R R R P/F S S S 0
RR	RR	R R R P/F 0 0 0 1
RNR	RNR	R R R P/F 0 1 0 1
SNRM		1 0 0 P 0 0 1 1
DISC		0 1 0 P 0 0 1 1
	UA	0 1 1 F 0 0 1 1
	DM	0 0 0 F 1 1 1 1
	FRMR	1 0 0 F 0 1 1 1
UI	UI	0 0 0 P/F 0 0 1 1

RRR is the receive sequence number N(R), **SSS** is the send sequence number N(S) and **P/F** is the poll/final bit.

NOTE In this notation, the bit order is inverted compared to the notation in ISO/IEC 13239. However, in both notations, the LSB is the first bit transmitted. For transmission order, see the complete Green Book.

For more information see the complete Green Book.

8.5 FCS calculation

8.5.1 Test sequence for the FCS calculation

NOTE The test sequence presented here can be found in the 1988 CCITT Blue Book X.1 – X.32, Appendix I.

The example presented here shows the proper FCS value for a two-byte frame consisting of 0x03 and 0x3F. The complete resulting frame is 0x7E 0x03 0x3F 0x5B 0xEC 0x7E.

```
\/ – first bit transmitted                                   last bit transmitted – \/
0111 1110 1100 0000     1111 1100     1101 1010 0011 0111       0111 1110
 flag          address        control                 FCS                   flag
```

In the test sequence, the following rules (according to ISO/IEC 13239) are considered:

- the FCS is calculated considering the bit order as transmitted on the channel;
- for the address field, the control field and all the other fields (including the data, except the FCS) the low order bit (of each byte) is transmitted first (this rule is automatically followed by the UART);
- for the FCS the coefficient of highest term (corresponding to x15) is transmitted first.

8.5.2 Fast frame check sequence (FCS) implementation

The following example implementation of the 16-bit FCS calculation is derived from RFC 1662.

The FCS was originally designed with hardware implementations in mind. A serial bit stream is transmitted on the wire, the FCS is calculated over the serial data as it goes out and the complement of the resulting FCS is appended to the serial stream, followed by the Flag Sequence.

The receiver has no way of determining that it has finished calculating the received FCS until it detects the Flag Sequence. Therefore, the FCS was designed so that a particular pattern results when the FCS operation passes over the complemented FCS. A good frame is indicated by this "good FCS" value.

For more information see the complete Green Book.

9 DLMS/COSEM application layer

9.1 DLMS/COSEM application layer main features

9.1.1 General

This subclause 9.1 provides an overview of the main features provided by the DLMS/COSEM AL.

9.1.2 DLMS/COSEM application layer structure

The structure of the client and server DLMS/COSEM application layers is shown in Figure 25.

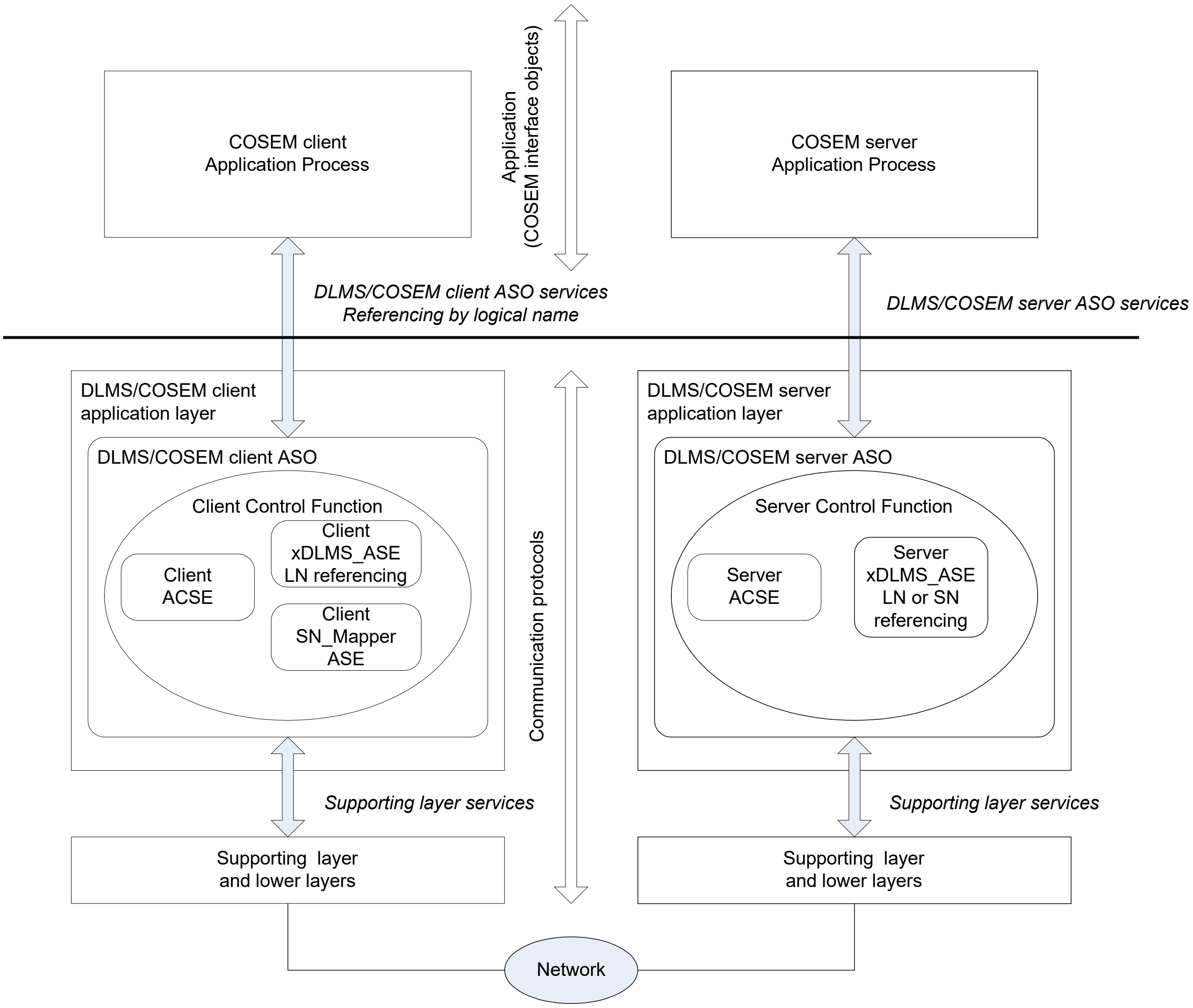


Figure 25 – The structure of the DLMS/COSEM application layers

The main component of the DLMS/COSEM AL is the Application Service Object (ASO). It provides services to its service user, the COSEM Application Process (APs) and uses services provided by the supporting layer. It contains three mandatory components both on the client and on the server side:

- the Association Control Service Element, ACSE;
- the extended DLMS Application Service Element, xDLMS ASE;
- the Control Function, CF.

On the client side, there is a fourth, optional element, called the Client SN_Mapper ASE.

The ACSE provides services to establish and release application associations (AAs). See 9.1.3.

The xDLMS ASE provides services to transport data between COSEM APs. See 9.1.4.

The Control Function (CF) element specifies how the ASO services invoke the appropriate service primitives of the ACSE, the xDLMS ASE and the services of the supporting layer. See also 9.4.1.

Both the client and the server DLMS/COSEM ASO may contain other, optional application protocol components.

The optional Client SN_Mapper ASE is present in the client side AL ASO, when the server uses SN referencing. It provides mapping between services using LN and SN referencing. See 9.1.5.

The DLMS/COSEM AL performs also some functions of the OSI presentation layer:

- encoding and decoding the ACSE APDUs and the xDLMS APDUs, see also 9.4.3;
- alternatively, generating and using XML documents representing ACSE and xDLMS APDUs;
- applying compression and decompression;
- applying, verifying and removing cryptographic protection.

9.1.3 The Association Control Service Element, ACSE

For the purposes of DLMS/COSEM connection oriented (CO) communication profiles, the CO ACSE, specified in ISO/IEC 15953:1999 and ISO/IEC 15954:1999 is used.

The services provided for application association establishment and release are the following:

- COSEM-OPEN;
- COSEM-RELEASE;
- COSEM-ABORT.

The COSEM-OPEN service is used to establish AAs. It is based on the ACSE A-ASSOCIATE service. It causes the start of use of an AA by those ASE procedures identified by the value of the Application_Context_Name, Security_Mechanism_Name and xDLMS context parameters. AAs may be established in different ways:

- confirmed AAs are established via a message exchange – using the COSEM-OPEN service – between the client and the server to negotiate the contexts. Confirmed AAs can be established between a single client and a single server;
- unconfirmed AAs are established via a message sent – using the COSEM-OPEN service – from the client to the server, using the parameters of the contexts supposed to be supported by the server. Unconfirmed AAs can be established between a client and one or multiple servers;
- pre-established AAs may pre-exist. In this case, the COSEM-OPEN service is not used. The client has to be aware of the contexts supported by the server. A pre-established AA can be confirmed or unconfirmed.

The COSEM-RELEASE service is used to release AAs. If successful, it causes the completion of the use of the AA without loss of information in transit (graceful release). In some communication profiles – for example in the TCP-UDP/IP based profile – the COSEM-RELEASE service is based on the ACSE A-RELEASE service. In some other communication profiles – for example in the 3-layer, CO, HDLC based profile – there is a one-to-one relationship between a confirmed AA and the supporting protocol layer connection. In such profiles AAs can be released simply by disconnecting the corresponding supporting layer connection. Pre-established AAs cannot be released.

The COSEM-ABORT service causes the abnormal release of an AA with the possible loss of information in transit. It does not rely on the ACSE A-ABORT service.

The COSEM-OPEN service is specified in 9.3.2, the COSEM-RELEASE service in 9.3.3 and the COSEM-ABORT service in 9.3.4.

9.1.4 The xDLMS application service element

9.1.4.1 Overview

To access attributes and methods of COSEM objects, the services of the xDLMS ASE are used. It is based on the DLMS standard, IEC 61334-4-41:1996. This Technical Report specifies a range of extensions to extend functionality while maintaining backward compatibility. The extensions comprise the following:

- additional services, see 9.1.4.3;
- additional mechanisms, see 9.1.4.4;
- additional data types, see 9.1.4.5;
- new DLMS version number, see 9.1.4.6;
- new conformance block, see 9.1.4.7;
- clarification of the meaning of the PDU size, see 9.1.4.8.

9.1.4.2 The xDLMS initiate service

To establish the xDLMS context the xDLMS Initiate service — specified in IEC 61334-4-41:1996, 5.2 — is used. This service is integrated in the COSEM-OPEN service, see 9.3.2.

9.1.4.3 COSEM object related xDLMS services

9.1.4.3.1 General

COSEM object related xDLMS services are used to access COSEM object attributes and methods.

DLMS UA 1000-1 Ed. 13:2019, 4.1.2 specifies two referencing methods:

- Logical Name (LN) referencing; and
- Short Name (SN) referencing.

For more information on referencing methods, see 9.1.4.4.2.

Therefore, two distinct xDLMS service sets are specified: one exclusively using Logical Name (LN) referencing and the other exclusively using short name (SN) referencing. It can be considered that there are two different xDLMS ASEs: one providing services with LN referencing and the other with SN referencing. The client ASO always uses the xDLMS ASE with LN referencing. The server ASO may use either the xDLMS ASE with LN referencing or the xDLMS ASE with SN referencing or both.

These services may be:

- requested / solicited by the client; or
- unsolicited: these are always initiated by the server without a previous request from the client.

Services requested by the client may be also (see 9.4.6.2):

- confirmed: in this case, the server provides a response to the request;
- unconfirmed: in this case, the server does not provide a response to the request.

The additional services – which are not based on DLMS services specified in IEC 61334-4-41:1996 – are:

- the GET, SET, ACTION and ACCESS used to access COSEM object attributes and methods using LN referencing;
- the DataNotification service used by the server to push data to the client;
- the EventNotification service used by the server to notify the client about events that occur in the server.

9.1.4.3.2 xDLMS services used by the client with LN referencing

In the case of LN referencing, COSEM object attributes and methods are referenced via the identifier of the COSEM object instance to which they belong. For this referencing method, the following additional services are specified:

- the GET service is used by the client to request the server to return the value of one or more attributes, see 9.3.6;
- the SET service is used by the client to request the server to replace the content of one or more attributes, see 9.3.7;
- the ACTION service is used by the client to request the server to invoke one or more methods. Invoking methods may imply sending method invocation parameters and receiving return parameters, see 9.3.8;
- the ACCESS service, a unified service which can be used by the client to access multiple attributes and/or methods with a single .request; see 9.3.9.

These services can be invoked by the client in a confirmed or unconfirmed manner.

9.1.4.3.3 xDLMS services used by the client with SN referencing

In the case of SN referencing, COSEM object attributes and methods are mapped to DLMS named variables specified in IEC 61334-4-41:1996, 10.1.2.

The xDLMS services using SN referencing are based on the DLMS variable access services, specified in IEC 61334-4-41:1996 subclauses 10.4 – 10.6 and they are the following:

- the Read service is used by the client to request the server to return the value of one or more attributes or to invoke one or more methods when return parameters are expected. It is a confirmed service. See 9.3.14;
- the Write service is used by the client to request the server to replace the content of one or more attributes or to invoke one or more methods when no return parameters are expected. It is a confirmed service. See 9.3.15;
- the UnconfirmedWrite service is used by the client to request the server to replace the content of one or more attributes or to invoke one or more methods when no return parameters are expected. It is an unconfirmed service. See 9.3.16.

New variants of the Variable_Access_Specification service parameter (see 9.3.13), the Read.response and the Write.response services have been added to support selective access – see 9.1.4.3.5 – and block transfer, see 9.1.4.4.5.

9.1.4.3.4 Unsolicited services

Unsolicited services are initiated by the server, on pre-defined conditions, e.g. schedules, triggers or events, to inform the client of the value of one or more attributes, as though they had been requested by the client.

To support push operation, the DataNotification service is available, see 9.3.10. It can be used in application contexts using either SN referencing or LN referencing.

NOTE The DataNotification service is used in conjunction with “Push setup” COSEM objects, see DLMS UA 1000-1 Ed. 13:2019, 4.4.8.

To support event notification, the following unsolicited services are available:

- with LN referencing, the EventNotification service, see 9.3.11;
- with SN referencing, the InformationReport service, see 9.3.17. This service is based on IEC 61334-4-41:1996, 10.7.

9.1.4.3.5 Selective access

In the case of some COSEM interface classes, selective access to attributes is available, meaning that either the whole attribute or a selected portion of it can be accessed as required. For this purpose, access selectors and parameters are specified as part of the specification of the relevant attributes.

To use this possibility, attribute-related services can be invoked with access selection parameters. In the case of LN referencing, this feature is called Selective access; see 9.3.6 and 9.3.7. It is a negotiable feature; see 9.4.6.1. In the case of SN referencing, this feature is called Parameterized access; see 9.3.14, 9.3.15 and 9.3.16. It is a negotiable feature; see 9.4.6.1.

9.1.4.3.6 Multiple references

In a COSEM object related service invocation, it is possible to reference one or several named variables, attributes and/or methods. Using multiple references is a negotiable feature. See 9.4.6.1.

9.1.4.3.7 Attribute_0 referencing

With the GET, SET and ACCESS services a special feature, Attribute_0 referencing is available. By convention, attributes of COSEM objects are numbered from 1 to n, where Attribute_1 is the logical name of the COSEM object. Attribute_0 has a special meaning: it references all attributes with positive index (public attributes). The use of Attribute_0 referencing with the GET service is explained in 9.3.6, with the SET service in 9.3.7 and with the ACCESS service in 9.3.9.

NOTE As specified in DLMS UA 1000-1 Ed. 13:2019, 4.1.2, manufacturers may add proprietary methods and/or attributes to any object, using negative numbers.

Attribute_0 referencing is a negotiable feature, see 9.4.6.1.

9.1.4.4 Additional mechanisms

9.1.4.4.1 Overview

xDLMS specifies several new mechanisms – compared to DLMS as specified in IEC 61334-4-41:1996 – to improve functionality, flexibility and efficiency. The additional mechanisms are:

- referencing using logical names;
- identification of service invocations;
- priority handling;
- transferring long application messages;
- composable xDLMS messages;
- compression and decompression;
- general cryptographic protection;
- general block transfer.

9.1.4.4.2 Referencing methods and service mapping

To access COSEM object attributes and methods with the xDLMS services, they have to be referenced. As already mentioned in 9.1.4.3.1, DLMS UA 1000-1 Ed. 13:2019, 4.1.2 specifies two referencing methods:

- Logical Name (LN) referencing; and
- Short Name (SN) referencing.

In the case of LN referencing, COSEM object attributes and methods are referenced via the logical name (COSEM_Object_Instance_ID) of the COSEM object instance to which they belong. In the case of SN referencing, COSEM object attributes and methods are mapped to DLMS named variables.

Accordingly, there are two xDLMS ASEs specified: one using xDLMS services with LN referencing and one using xDLMS services with SN referencing.

On the client side, in order to handle the different referencing methods transparently for the AP, the AL uses the xDLMS ASE with LN referencing. Using a unique, standardized service set between COSEM client APs and the communication protocol – hiding the particularities of DLMS/COSEM servers using different referencing methods – allows specifying an Application Programming Interface, API. This is an explicitly specified interface corresponding to this service set for applications running in a given computing environment (for example Windows, UNIX, etc.) Using this – public – API specification, client applications can be developed without knowledge about particularities of a given server.

On the server side, either the xDLMS ASE with LN referencing or the xDLMS ASE with SN referencing or both xDLMS ASEs can be used.

In the case of confirmed AAs, the referencing method is negotiated during the AA establishment phase via the COSEM application context. It shall not change during the lifetime of the AA established. Using LN or SN services within a given AA is exclusive.

In the case of unconfirmed and pre-established AAs, the client AL is expected to know the referencing method supported by the server.

When the server uses LN referencing, the services are the same on both sides. When the server uses SN referencing the Client SN_Mapper ASE in the client maps the SN referencing into LN referencing or vice versa. See 9.1.2 and 9.1.5.

9.1.4.4.3 Identification of service invocations: the Invoke_Id parameter

In the client/server model, requests are sent by the client and responses are sent by the server. The client is allowed to send several requests before receiving the response to the previous ones, provided that this is allowed by the lower layers.

Therefore – to be able to identify which response corresponds to each request – it is necessary to include a reference in the request.

The Invoke_Id parameter is used for this purpose. The value of this parameter is assigned by the client so that each request carries a different Invoke_Id. The server shall copy the Invoke_Id into the corresponding response.

In the ACCESS and the DataNotification service – see 9.3.9 and 9.3.10 – the Long-Invoke-Id parameter is used instead of the Invoke_Id parameter.

The EventNotification service does not contain the Invoke_Id parameter.

This feature is available only with LN referencing.

9.1.4.4.4 Priority handling

For data transfer services using LN referencing, two priority levels are available: normal (FALSE) and high (TRUE). This feature allows receiving a response to a new request before the response to a previous request is completed.

Normally, the server serves incoming service requests in the order of reception (FIFS, First In, First Served). However, a request with the priority parameter set to high (TRUE) is served before the previous requests with priority set to normal (FALSE). The response carries the same priority flag as that of the corresponding request. Managing priority is a negotiable feature; see 9.4.6.1.

NOTE 1 As service invocations are identified with an Invoke_Id, services with the same priority can be served in any order.

NOTE 2 If the feature is not supported, requests with HIGH priority are served with NORMAL priority.

This feature is not available with services using SN referencing. The server treats the services on a FIFS basis.

9.1.4.4.5 Transferring long messages

The xDLMS service primitives are carried in an encoded form by xDLMS APDUs. This encoded form may be longer than the Client / Server Max Receive PDU Size negotiated. To transfer such 'long' messages, there are two mechanisms available:

a) the general block transfer (GBT) mechanism specified in 9.1.4.4.9;

b) the service-specific block transfer mechanism.

Using the general or the service-specific block transfer mechanism is a negotiable feature; see 9.4.6.1.

An APDU that fits in the Client / Server Max Receive PDU Size negotiated may be too long to fit in a single frame / packet of the supporting layer. Such APDUs may be transported if the supporting layer provide(s) segmentation; see Clause 10.

Service-specific block transfer mechanism is available with:

- confirmed services using LN referencing: GET, SET, ACTION;
- confirmed services using SN referencing: Read, Write.

and the related APDUs.

Service-specific block transfer is not available with:

- unconfirmed services;
- unsolicited services (DataNotification, EventNotification and InformationReport);
- the ACCESS service

and the related APDUs.

With service-specific block transfer, the reception of each block has to be acknowledged before the next block can be sent. A recovery mechanism for lost blocks is not available. When global key or dedicated key ciphering is required, it is applied to the APDU carrying a block – or an acknowledgement of a block – of the long message. This creates considerable computing and transport overhead. Service specific digital signature is not available.

Conversely, the GBT mechanism can be used with any xDLMS APDU including the general-ciphering and general-signing APDUs. It provides bidirectional block transfer, streaming and lost block recovery. When cryptographic protection is required, it is applied to the complete APDU and then the protected APDU is transferred in blocks, as specified in 9.1.4.4.6 Figure 26.

9.1.4.4.6 Composable xDLMS messages

The three important aspects of dealing with xDLMS messages are encoding / decoding, applying, verifying / removing cryptographic protection and block transfer.

The concept of composable xDLMS messages separates the three aspects, as shown in Figure 26. See also Figure 36.

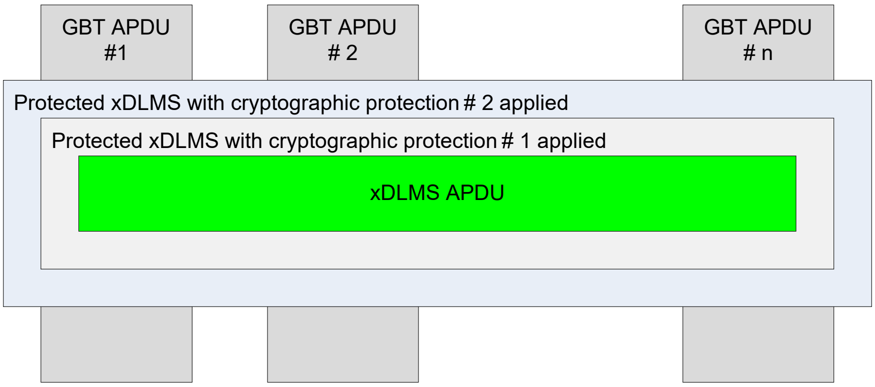

GBT APDU #1
GBT APDU # 2
GBT APDU # n
Protected xDLMS with cryptographic protection # 2 applied
Protected xDLMS with cryptographic protection # 1 applied
xDLMS APDU

Figure 26 – The concept of composable xDLMS messages

Once the APDU corresponding to the service primitive invoked by the AP is built by the AL, the general protection mechanism can be used to apply cryptographic protection. When an unprotected or a protected APDU is too long to fit in the negotiated APDU size, then the general block transfer mechanism can be applied.

These mechanisms can be applied with all xDLMS APDUs.

NOTE 1 With the GET, SET, ACTION, EventNotification, Read and Write, UnconfirmedWrite and InformationReport services, service-specific cryptographic protection is available using specific service protection types and APDUs.

NOTE 2 With the GET, SET, ACTION, Read, Write, and UnconfirmedWrite and services, service-specific block transfer is available using specific service request / response types and APDUs.

9.1.4.4.7 Compression and decompression

In order to optimize the use of communication media, it is possible to compress xDLMS APDUs to be sent and decompress xDLMS APDUs received. For details, see the complete Green Book.

9.1.4.4.8 General protection

This mechanism can be used to apply cryptographic protection to any xDLMS APDU and this allows applying multiple layers of protection between the client and the server or between a third party and the server. See also the complete Green Book.

For this purpose, the following APDUs are available; see 9.2.7.2.3:

- the general-ded-ciphering and the general-glo-ciphering APDUs;
- the general-ciphering APDUs;
- the general-signing APDU.

Using the general protection mechanism is a negotiable feature, see 9.4.6.1.

9.1.4.4.9 General block transfer (GBT)

The GBT mechanism can be used to transfer any – short or long – xDLMS APDU in blocks. With GBT, the blocks are carried by general-block-transfer APDUs instead of service-specific "with-datablock" APDUs.

The GBT mechanism supports bi-directional block transfer, streaming and lost block recovery:

- bi-directional block transfer means that while one party is sending blocks, the other party not only confirms the blocks received but if it has blocks to send it can send them as well while it is still receiving blocks;

  NOTE Bi-directional block transfer is useful when long service parameters need to be transported in both directions.

- streaming means that several blocks may be sent – streamed – by one party without an acknowledgement of each block from the other party;
- lost block recovery means that if the reception of a block sent is not confirmed, it can be sent again. If streaming is used, lost block recovery takes place at the end of each streaming window.

The GBT mechanism is managed by the AL using the block transfer streaming parameters specified in 9.3.5.

The protocol of the general block transfer mechanism is specified in 9.4.6.13

The GBT mechanism supports the following use cases:

1) short request by the client that leads to a long response from the server: the request may be sent with or without using GBT. If it is sent using GBT then the client may advertise its preferred GBT window size when the exchange is started. The server responds using GBT;
2) long request by the client: the client sends the request and the server sends the response using GBT, whatever is the length of the response;
3) long unconfirmed request by the client: the client sends the request using GBT;
4) long unsolicited request from the server: the server sends the unsolicited message using GBT.

9.1.4.5 Additional data types

The additional data types are specified in 9.5 and in 9.6.

9.1.4.6 xDLMS version number

The new DLMS version number, corresponding to the first version of the xDLMS ASE is 6.

9.1.4.7 xDLMS conformance block

The xDLMS conformance block enables optimised DLMS/COSEM server implementations with extended functionality. It can be distinguished from the DLMS conformance block by its tag "Application 31". See 9.4.6.1, 9.5 and in 9.6.

The xDLMS conformance block is part of the xDLMS context.

In the case of confirmed AAs, the conformance block is negotiated during the AA establishment phase via the xDLMS context. It shall not change during the lifetime of the AA established.

In the case of unconfirmed and pre-established AAs, the client AL is expected to know the conformance block supported by the server.

9.1.4.8 Maximum PDU size

To clarify the meaning of the maximum PDU size usable by the client and the server, the modifications shown in Table 6 have been made. The xDLMS Initiate service uses these names for PDU sizes.

Table 6 – Clarification of the meaning of PDU size for DLMS/COSEM

was:	new:
IEC 61334-4-41:1996, 5.2.2, Table 3	
Proposed Max PDU Size	Client Max Receive PDU Size
Negotiated Max PDU Size	Server Max Receive PDU Size
IEC 61334-4-41:1996, 5.2.3, 7th paragraph	
The Proposed Max PDU Size parameter, of type Unsigned16, proposes a maximum length expressed in bytes for the exchanged DLMS APDUs. The value proposed in an Initiate request must be large enough to always permit the Initiate Error PDU transmission	The Client Max Receive PDU Size parameter, of type Unsigned16, contains the maximum length expressed in bytes for a DLMS APDU that the server may send. The client will discard any received PDUs that are longer than this maximum length. The value must be large enough to always permit the AARE APDU transmission. Values below 12 are reserved. The value 0 indicates that there is no limit on the PDU size.
IEC 61334-4-41:1996, 5.2.3, last paragraph	
The Negotiated Max PDU Size parameter, of type Unsigned16, contains a maximum length expressed in bytes for the exchanged DLMS APDUs. A PDU that is longer than this maximum length will be discarded. This maximum length is computed as the minimum of the Proposed Max PDU Size and the maximum PDU size than the VDE-handler may support.	The Server Max Receive PDU Size parameter, of type Unsigned16, contains the maximum length expressed in bytes for a DLMS APDU that the client may send. The server will discard any received PDUs that are longer than this maximum length. Values below 12 are reserved. The value 0 indicates that there is no limit on the PDU size.

9.1.5 Layer management services

Layer management services have local importance only. Therefore, specification of these services is not within the Scope of this Technical Report.

The specific SetMapperTable service is defined in the complete Green Book.

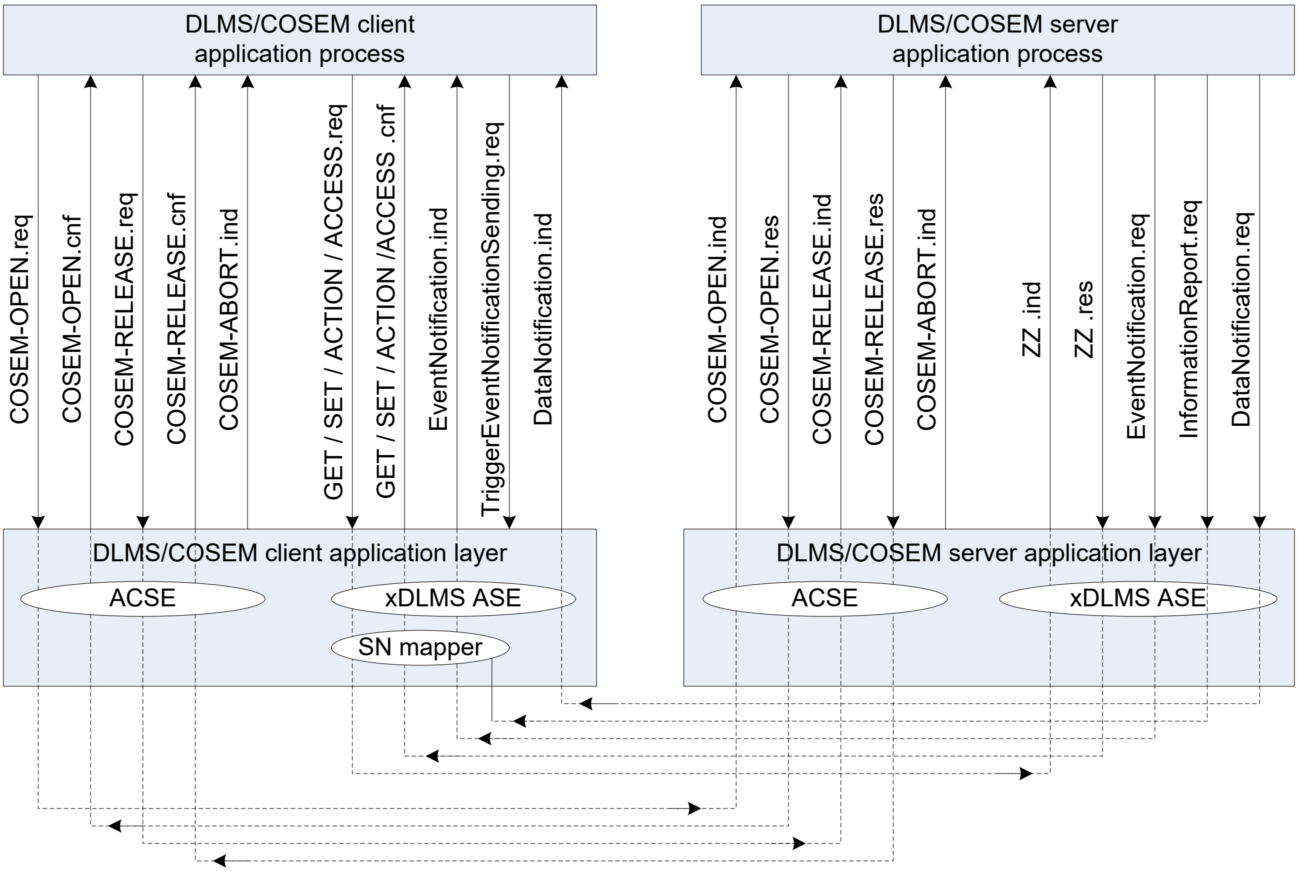
9.1.6 Summary of DLMS/COSEM application layer services

A summary of the services available at the top of the DLMS/COSEM AL is shown in Figure 27. Layer management services are not shown. Although the service primitives are different on the client and server side, the APDUs are the same.

NOTE 1 For example, when the client AP invokes a GET.request service primitive the client AL builds a GET-Request APDU. When this is received by the server AL, it invokes a GET.ind service primitive.

The DLMS/COSEM AL services are specified in 9.3. The DLMS/COSEM AL protocol is specified in 9.4. The abstract syntax of the ACSE and xDLMS APDUs is specified in 9.5. The XML schema is defined in 9.6.

Encoding examples are provided in Clauses 11, 12, 13 and 14.



NOTE 2 The client AP always uses LN referencing. If the server uses SN referencing then a mapping is performed by the Client SN_Mapper ASE. Consequently, the service primitives ZZ.ind and ZZ.res may be LN or SN service primitives. LN/SN service mapping is specified in 9.5.

NOTE 3 The ACCESS service cannot be mapped to services using SN referencing.

Figure 27 – Summary of DLMS/COSEM AL services

9.1.7 DLMS/COSEM application layer protocols

The DLMS/COSEM AL protocols specify the procedures for information transfer for AA control and authentication using connection-oriented ACSE procedures, and for data transfer between COSEM clients and servers using xDLMS procedures. Therefore, the DLMS/COSEM AL protocol is based on the ACSE standard as specified in ISO/IEC 15954:1999 and the DLMS standard, as specified in IEC 61334-4-41:1996, with the extensions for DLMS/COSEM. The procedures are defined in terms of:

- the interactions between peer ACSE and xDLMS protocol machines through the use of services of the supporting protocol layer;
- the interactions between the ACSE and xDLMS protocol machines and their service user.

The DLMS/COSEM AL protocols are specified in 9.4.

9.2 Information security in DLMS/COSEM

9.2.1 Overview

This subclause 9.2 describes and specifies:

- the DLMS/COSEM security concept, see 9.2.2;
- the cryptographic algorithms selected, see 9.2.3;
- the security keys, see 9.2.4, 9.2.5 and 9.2.6;
- the use of the cryptographic algorithms for entity authentication, xDLMS APDU protection and COSEM data protection, see 9.2.7.

9.2.2 The DLMS/COSEM security concept

9.2.2.1 Overview

The resources of DLMS/COSEM servers – COSEM object attributes and methods – can be accessed by DLMS/COSEM clients within Application Associations, see also 4.5.

During an AA establishment the client and the server have to identify themselves. The server may also require that the *user* of a client identifies itself. Furthermore, the server may require that the client authenticates itself and the client may also require that the server authenticates itself. The identification and authentication mechanisms are specified in 9.2.2.2.

Once an AA is established, xDLMS services can be used to access COSEM object attributes and methods, subject to the security context and access rights. See the complete Green Book.

The xDLMS APDUs carrying the services primitives can be cryptographically protected. The required protection is determined by the security context and the access rights. To support end-to-end security between third parties and servers, such third parties can also access the resources of a server using a client as a broker. The concept of message protection is further explained in the complete Green Book.

Moreover, COSEM data carried by the xDLMS APDUs can be cryptographically protected; see the complete Green Book.

As these security mechanisms are applied on the application process / application layer level, they can be used in all DLMS/COSEM communication profiles.

NOTE Lower layers may provide additional security.

9.2.2.2 Identification and authentication

9.2.2.2.1 Identification

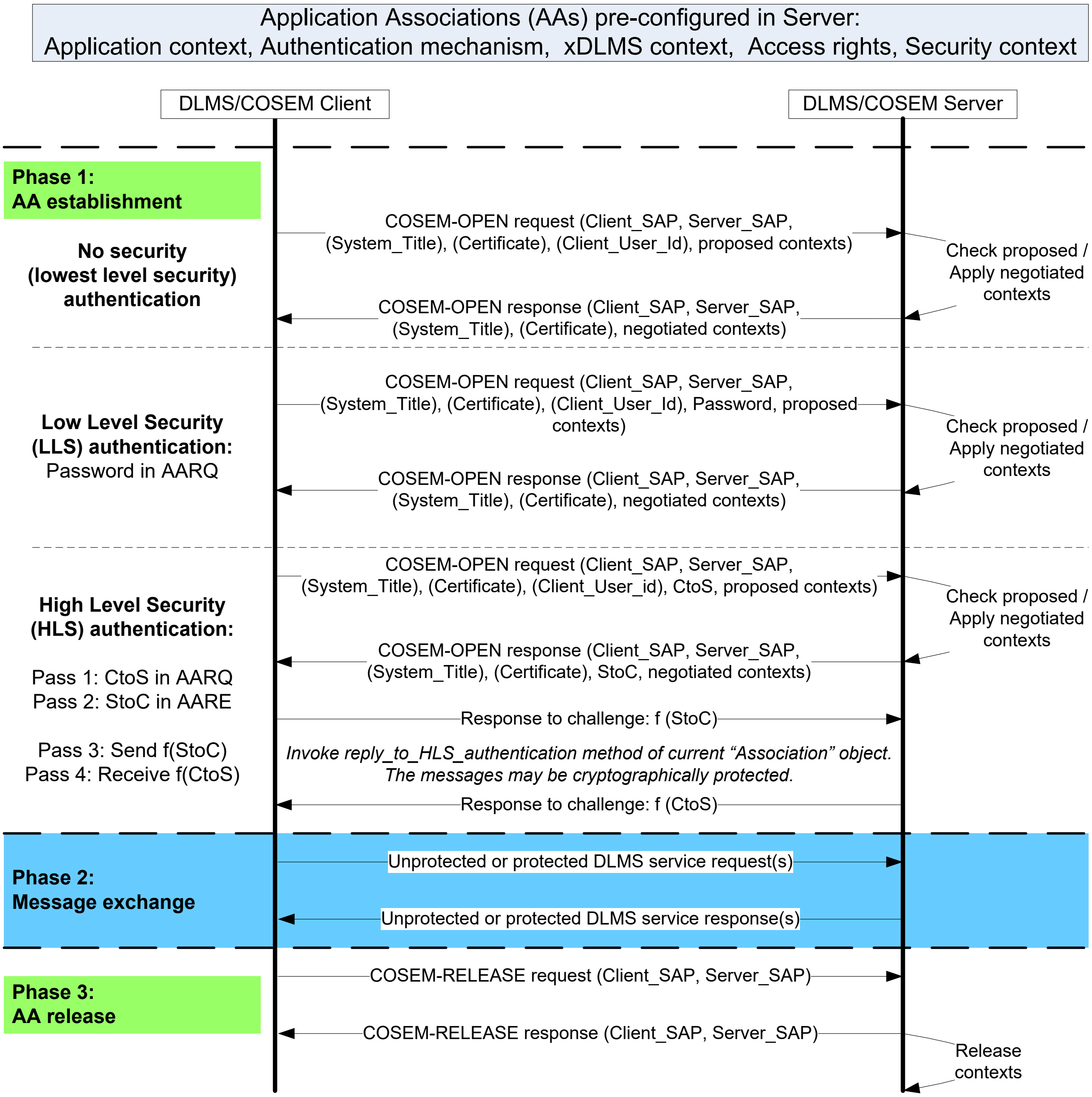
As specified in 4.3.3, DLMS/COSEM AEs are bound to Service Access Points (SAPs) in the protocol layer supporting the AL. These SAPs are present in the PDUs carrying the xDLMS APDUs within an AA.

The client user identification mechanism enables the server to distinguish between different users on the client side — that may be operators or third parties — to log their activities accessing the meter. See also 4.3.6.

9.2.2.2.2 Authentication mechanisms

9.2.2.2.2.1 Overview

The authentication mechanisms determine the protocol to be used by the communication entities to authenticate themselves during AA establishment.


Application Associations (AAs) pre-configured in Server:
Application context, Authentication mechanism, xDLMS context, Access rights, Security context
DLMS/COSEM Client
DLMS/COSEM Server
Phase 1:
AA establishment
No security
(lowest level security)
authentication
COSEM-OPEN request (Client_SAP, Server_SAP, (System_Title), (Certificate), (Client_User_Id), proposed contexts)
Check proposed / Apply negotiated contexts
COSEM-OPEN response (Client_SAP, Server_SAP, (System_Title), (Certificate), negotiated contexts)
Low Level Security
(LLS) authentication:
Password in AARQ
COSEM-OPEN request (Client_SAP, Server_SAP, (System_Title), (Certificate), (Client_User_Id), Password, proposed contexts)
Check proposed / Apply negotiated contexts
COSEM-OPEN response (Client_SAP, Server_SAP, (System_Title), (Certificate), negotiated contexts)
High Level Security
(HLS) authentication:
Pass 1: CtoS in AARQ
Pass 2: StoC in AARE
Pass 3: Send f(StoC)
Pass 4: Receive f(CtoS)
COSEM-OPEN request (Client_SAP, Server_SAP, (System_Title), (Certificate), (Client_User_id), CtoS, proposed contexts)
Check proposed / Apply negotiated contexts
COSEM-OPEN response (Client_SAP, Server_SAP, (System_Title), (Certificate), StoC, negotiated contexts)
Response to challenge: f (StoC)
Invoke reply_to_HLS_authentication method of current “Association” object.
The messages may be cryptographically protected.
Response to challenge: f (CtoS)
Phase 2:
Message exchange
Unprotected or protected DLMS service request(s)
Unprotected or protected DLMS service response(s)
Phase 3:
AA release
COSEM-RELEASE request (Client_SAP, Server_SAP)
COSEM-RELEASE response (Client_SAP, Server_SAP)
Release contexts

NOTE 1 The COSEM-OPEN service primitives are carried by AARQ / AARE APDUs. The COSEM-RELEASE service primitives are carried by RLRQ / RLRE APDUs (when used). See 9.3.2 and 9.3.3.

NOTE 2 The elements (System_Title), (Certificate) and (Client_User_Id) are optional.

NOTE 3 In pre-established AAs no authentication takes place.

NOTE 4 The COSEM-RELEASE service can be cryptographically protected by including a ciphered xDLMS Initiate .request / .response APDU in the RLRQ.

Figure 28 – Authentication mechanisms

There are three different authentication mechanisms available with different authentication security levels:

- no security (Lowest Level Security) authentication; see 9.2.2.2.2.2;
- Low Level Security (LLS) authentication, see 9.2.2.2.2.3;

  NOTE 1 In ITU-T X.811:1995 this is known as unilateral authentication, class 0 mechanism.

- High Level Security (HLS) authentication, see 9.2.2.2.2.4.

  NOTE 2 In ITU-T X.811:1995 this is known as mutual authentication using challenge mechanisms.

They are shown in Figure 28. Authentication mechanisms are identified by names, see 9.4.2.2.3

The security of the message exchange (in Phase 2) is independent of the client-server authentication during AA establishment (Phase 1). Even in the case where no client-server authentication takes place, cryptographically protected APDUs can be used to ensure message security.

9.2.2.2.2.2 No security (Lowest Level Security) authentication

The purpose of No security (Lowest Level Security) authentication is to allow the client to retrieve some basic information from the server. This authentication mechanism does not require any authentication; the client can access the COSEM object attributes and methods within the security context and access rights prevailing in the given AA.

9.2.2.2.2.3 Low Level Security (LLS) authentication

In this case, the server requires that the client authenticates itself by supplying a password that is known by the server. The password is held by the current “Association SN / LN” object modelling the AA to be established. The “Association SN / LN” objects provide means to change the secret.

If the password supplied is accepted, the AA can be established, otherwise it shall be rejected.

LLS authentication is supported by the COSEM-OPEN service – see 9.3.2 – as follows:

- the client transmits a “secret” (a password) to the server, using the COSEM-OPEN.request service primitive;
- the server checks if the “secret” is correct;
- if yes, the client is authenticated and the AA can be established. From this moment, the negotiated contexts are valid;
- if not, the AA shall be rejected;
- the result of establishing the AA shall be sent back by the server using the COSEM-OPEN.response service primitive, together with diagnostic information.

9.2.2.2.2.4 High Level Security (HLS) authentication

In this case, both the client and the server have to successfully authenticate themselves to establish an AA. HLS authentication is a four-pass process that is supported by the COSEM-OPEN service and the *reply_to_HLS_authentication* method of the “Association SN / LN” interface class:

- Pass 1: The client transmits a “challenge” *CtoS* and – depending on the authentication mechanism – additional information to the server;
- Pass 2: The server transmits a “challenge” *StoC* and – depending on the authentication mechanism – additional information to the client;

If *StoC* is the same as *CtoS*, the client shall reject it and shall abort the AA establishment process.

- Pass 3: The client processes *StoC* and the additional information according to the rules of the HLS authentication mechanism valid for the given AA and sends the result to the server. The server checks if f(*StoC*) is the result of correct processing and – if so – it accepts the authentication of the client;

- Pass 4: The server processes then *CtoS* and the additional information according to the rules of the HLS authentication mechanism valid for the given AA and sends the result to the client. The client checks if f(*CtoS*) is the result of correct processing and – if so – it accepts the authentication of the server.

Pass 1 and Pass 2 are supported by the COSEM-OPEN service.

After Pass 2 – provided that the proposed application context and xDLMS context are acceptable – the server grants access to the method reply_to_HLS_authentication of the current "Association SN / LN" object using the application context negotiated.

Pass 3 and Pass 4 are supported by the method reply_to_HLS_authentication of the "Association SN / LN" object(s). If both passes 3 and 4 are successfully executed, then the AA is established with the application context and xDLMS context negotiated.

The dedicated-key, if transferred, can be used from this moment.

Otherwise, either the client or the server aborts.

There are several HLS authentication mechanisms available. These are further specified in the complete Green Book.

In some HLS authentication mechanisms, the processing of the challenges involves the use of an HLS secret.

The "Association SN / LN" interface class provides a method to change the HLS "secret": *change_HLS_secret*.

For more information see the complete Green Book.

9.2.3 Cryptographic algorithms

For more information see the complete Green Book.

9.2.4 Cryptographic keys – overview

For more information see the complete Green Book.

9.2.5 Key used with symmetric key algorithms

For more information see the complete Green Book.

9.2.6 Keys used with public key algorithms

For more information see the complete Green Book.

9.2.7 Applying cryptographic protection

9.2.7.1 Overview

The cryptographic algorithms specified in 9.2.3 can be applied:

- to protect the xDLMS APDUs see 9.2.7.2;
- to process the challenges during HLS authentication, see the complete Green Book;
- to protect COSEM data, see the complete Green Book.

9.2.7.2 Protecting xDLMS APDUs

9.2.7.2.1 Overview

This subclause 9.2.7.2 specifies how the cryptographic algorithms specified in the complete Green Book can be used to protect xDLMS APDUs:

- 9.2.7.2.2 specifies the possible values of security policy and access rights;
- 9.2.7.2.3 presents the types of ciphered APDUs;
- 9.2.7.2.4 specifies the use of the AES-GCM algorithm for authentication and encryption;
- 9.2.7.2.5 specifies the use of the ECDSA algorithm for digital signature.

When a COSEM object attribute or method related xDLMS service primitive is invoked by the AP, the service parameters include the Security_Options parameter. This parameter informs the AL on the kind of ciphered APDU to be used, on the protection to be applied, and includes the necessary security material. The AL builds first the APDU corresponding to the service primitive then it builds the ciphered APDU.

When the AL receives a ciphered APDU from a remote partner, it deciphers it and restores the original, unciphered APDU. When this is successfully done, it invokes the appropriate service primitive. The additional service parameters include the Security_Status and the Protection_Element parameters that inform the AP about the kind of ciphered APDU used, on the protection that has been verified and removed, and may include security material. See also 9.3.5.

9.2.7.2.2 Security policy and access rights values

In the case of “Security setup” version 1 – see DLMS UA 1000-1 Ed. 13:2019, 4.4.7 – the enum type shall be interpreted as an unsigned 8 type; the meaning of each bit is as shown in Table 7.

Table 7 – Security policy values (“Security setup” version 1)

Bit	Security policy
0	unused, shall be set to 0
1	unused, shall be set to 0
2	authenticated request
3	encrypted request
4	digitally signed request
5	authenticated response
6	encrypted response
7	digitally signed response
NOTE In the case of “Security policy” version 0 the possible security policy values are specified in DLMS UA 1000-1 Ed. 13:2019, 5.4.8. For both “Security policy” version 0 and 1 the value (0) means that no cryptographic protection is required.	

Access rights are held by the *object_list* attribute of “Association LN” or the *access_rights_list* of “Association SN” objects. The access_mode element of the access_rights determines the access kind and stipulates the cryptographic protection. It is an *enum* data type.

In the case of “Association LN” version 3 and “Association SN” version 4 – see DLMS UA 1000-1 Ed. 13:2019, 4.4.4. and 4.4.3 – the *enum* value is interpreted as an *unsigned8* and the meaning of each bit is as shown in Table 8.

For older versions, see their specification in DLMS UA 1000-1, the “Blue Book”.

Table 8 – Access rights values (“Association LN” ver 3 “Association SN” ver 4)

Bit	Attribute access	Method access
0	read-access	access
1	write-access	not-used
2	authenticated request	authenticated request
3	encrypted request	encrypted request
4	digitally signed request	digitally signed request
5	authenticated response	authenticated response
6	encrypted response	encrypted response
7	digitally signed response	digitally signed response
Examples	enum (3): read-write enum (6) write access with authenticated request enum (255): read-write access with authenticated, encrypted and digitally signed request and response	enum (1): access enum (2): not used enum (5): access with authenticated request enum (253): access with authenticated, encrypted and digitally signed request and response

Access rights to COSEM object attributes and/or methods may require authenticated, encrypted and / or signed xDLMS APDUs. For this reason, APDUs with more protection than required by the security policy are always allowed. APDUs with less protection than required by the security policy and the access rights shall be rejected.
More protection in this context means that more kinds of protection are applied on the xDLMS APDU than what is requested by the security policy: for example, security policy requires that all APDUs are authenticated but access rights require that the APDU is enrcypted and authenticated i.e. a higher protection.

9.2.7.2.3 Ciphered xDLMS APDUs

The different kind of ciphered xDLMS APDUs are shown in Table 9. See also 9.3.5.

Ciphered xDLMS APDUs can be used in a ciphered application context only. On the other hand, in a ciphered application context, both ciphered and unciphered APDUs may be used.

Table 9 – Ciphered xDLMS APDUs

APDU type	Parties	Type of ciphering	Security services	Key used	Compression
Service-specific glo-ciphering or ded-ciphering	Client – Server	Symmetric key	Authentication Encryption	Block cipher key: - Dedicated key [1] - Global unicast / broadcast key *established* [2] outside the exchange [3], *identified* by the SC byte Authentication key: global, *established* [2] outside the exchange [3]	–
general-glo-ciphering general-ded-ciphering					Yes [5]
general-ciphering	Third party or Client – Server	Symmetric key	Authentication Encryption	Block cipher key: - Global unicast / broadcast, established [2] outside the exchange [3], identified as part of the exchange, or - Established [2] as part of the exchange [4] Authentication key: global, established [2] outside the exchange [3]	Yes [5]
general-signing		Asymmetric key	Digital signature	Signing key	No

[1] Transported by the AARQ;

[2] Key establishment may be key wrapping or key agreement;

[3] In the server, these keys are held by the Security setup objects;

[4] Key data is transported in the protected APDU;

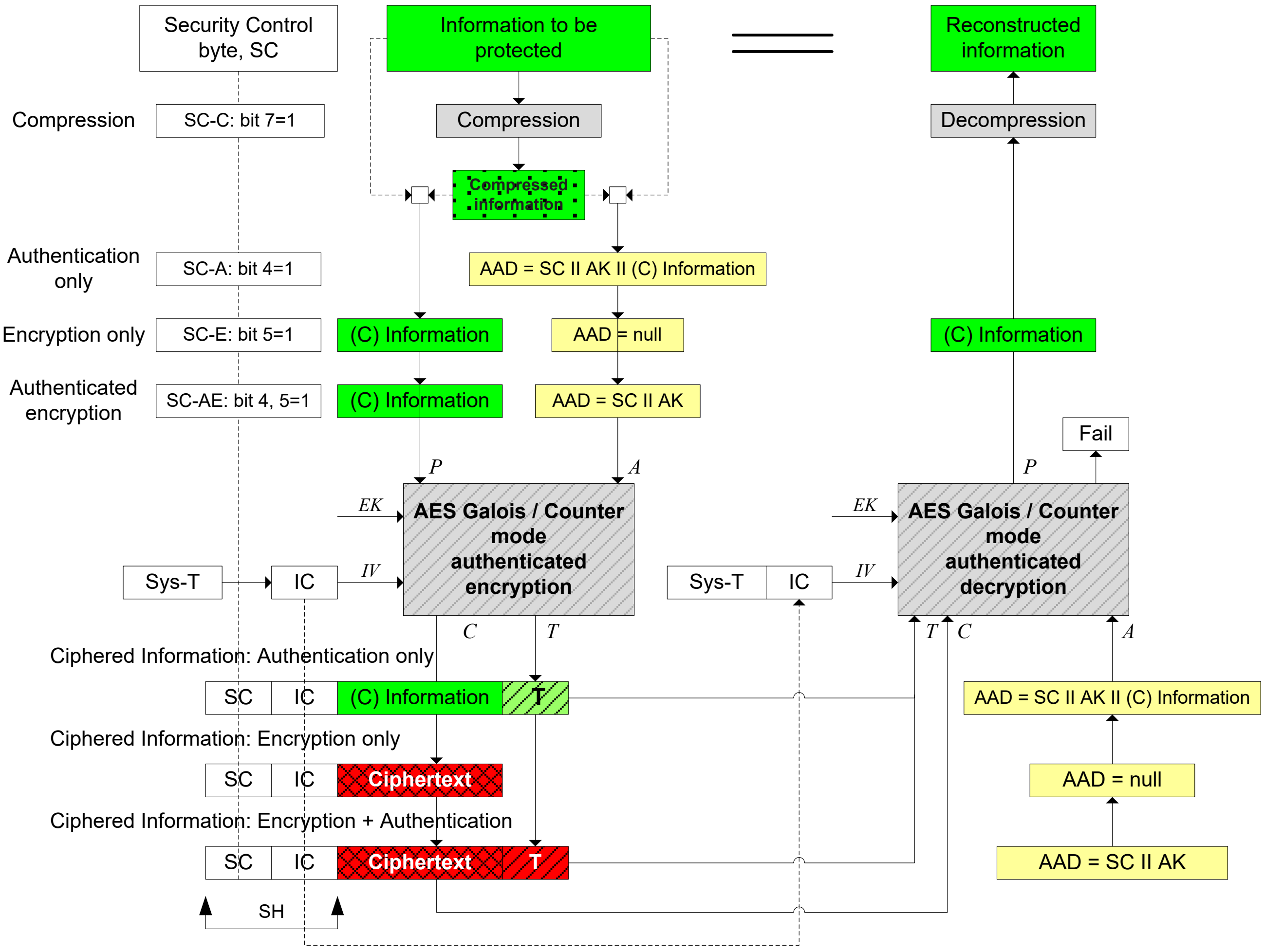
[5] The use of compression is controlled by the Security Control byte.

9.2.7.2.4 Encryption, authentication and compression

9.2.7.2.4.1 Overview

Encryption and authentication to protect information using the AES-GCM algorithm is shown in Figure 29. See also the complete Green Book. This algorithm can be combined with compression.

In the case of message protection, the information to be protected is an xDLMS APDU. In the case of COSEM data protection, the information to be protected is COSEM data, i.e. COSEM attribute value(s) or method invocation / return parameter(s).


Security Control byte, SC
Information to be protected
Reconstructed information
Compression
SC-C: bit 7=1
Compression
Decompression
Compressed information
Authentication only
SC-A: bit 4=1
AAD = SC II AK II (C) Information
Encryption only
SC-E: bit 5=1
(C) Information
AAD = null
(C) Information
Authenticated encryption
SC-AE: bit 4, 5=1
(C) Information
AAD = SC II AK
Fail
P
A
P
EK
AES Galois / Counter mode authenticated encryption
EK
AES Galois / Counter mode authenticated decryption
Sys-T
IC
IV
Sys-T
IC
IV
C
T
T
C
A
Ciphered Information: Authentication only
SC
IC
(C) Information
T
AAD = SC II AK II (C) Information
Ciphered Information: Encryption only
SC
IC
Ciphertext
AAD = null
Ciphered Information: Encryption + Authentication
SC
IC
Ciphertext
T
AAD = SC II AK
SH

Additional Authenticated Data AAD (Associated data) contains:
- Authentication only: SC II AK II (C) Information;
- Encryption only: Null
- Authenticated encryption: SC II AK
- In the case of the general-ciphering service, AAD contains additional elements as well.

(C) Information: Compressed information

A = AAD	*P* = Plaintext
AK =Authentication key	SC = Security control byte
C = Ciphertext	SH = SC II IC Security header
EK = Encryption key	*Sys-T* = System title (originator)
IC = Invocation counter	*T* = Authentication tag
IV = Sys-T II IC Initialization vector	

NOTE In the case of general-ciphering, AAD also includes additional fields, see the complete Green Book.

Figure 29 – Cryptographic protection of information using AES-GCM

The security material required is specified in 9.2.7.2.4.2 – 9.2.7.2.4.5.

9.2.7.2.4.2 The security header

The security header SH includes the security control byte concatenated with the invocation counter: SH = SC II IC.

For more information see the complete Green Book.

9.2.7.2.4.3 Plaintext and Additional Authenticated Data

The plaintext denoted *P* and the Additional Authenticated Data denoted *A* depend on the kind of protection.

For more information see the complete Green Book.

9.2.7.2.4.4 Encryption key and authentication key

These keys used by AES-GCM are specified in the complete Green Book. The various keys used in DLMS/COSEM and their establishment are specified in 9.2.5.

9.2.7.2.4.5 Initialization vector

For more information see the complete Green Book.

9.2.7.2.4.6 Service-specific ciphering xDLMS APDUs

For certain xDLMS APDUs – see 9.1.4.4.6 and 9.5 – a ciphered variant using the global key and one using the dedicated key is available. These ciphered APDUs can be used between a client and a server. The structure of the service-specific ciphering APDUs is shown in Figure 30. See also the complete Green Book.

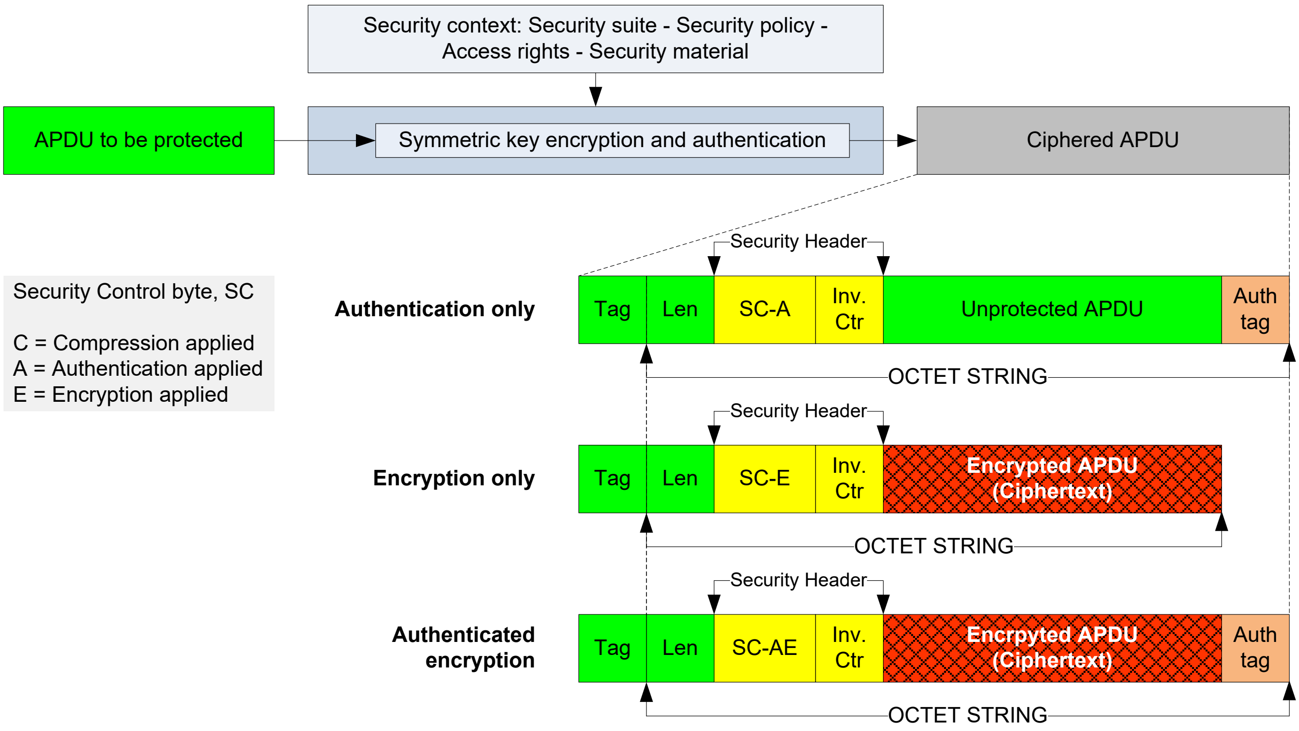


Figure 30 – Structure of service-specific global / dedicated ciphering xDLMS APDUs

9.2.7.2.4.7 The general-glo-ciphering and general-ded-ciphering xDLMS APDUs

These APDUs can be used to cipher other xDLMS APDUs using the global key or the dedicated key. They can be used between a client and a server. Their structure is shown in Figure 31. See also the complete Green Book.

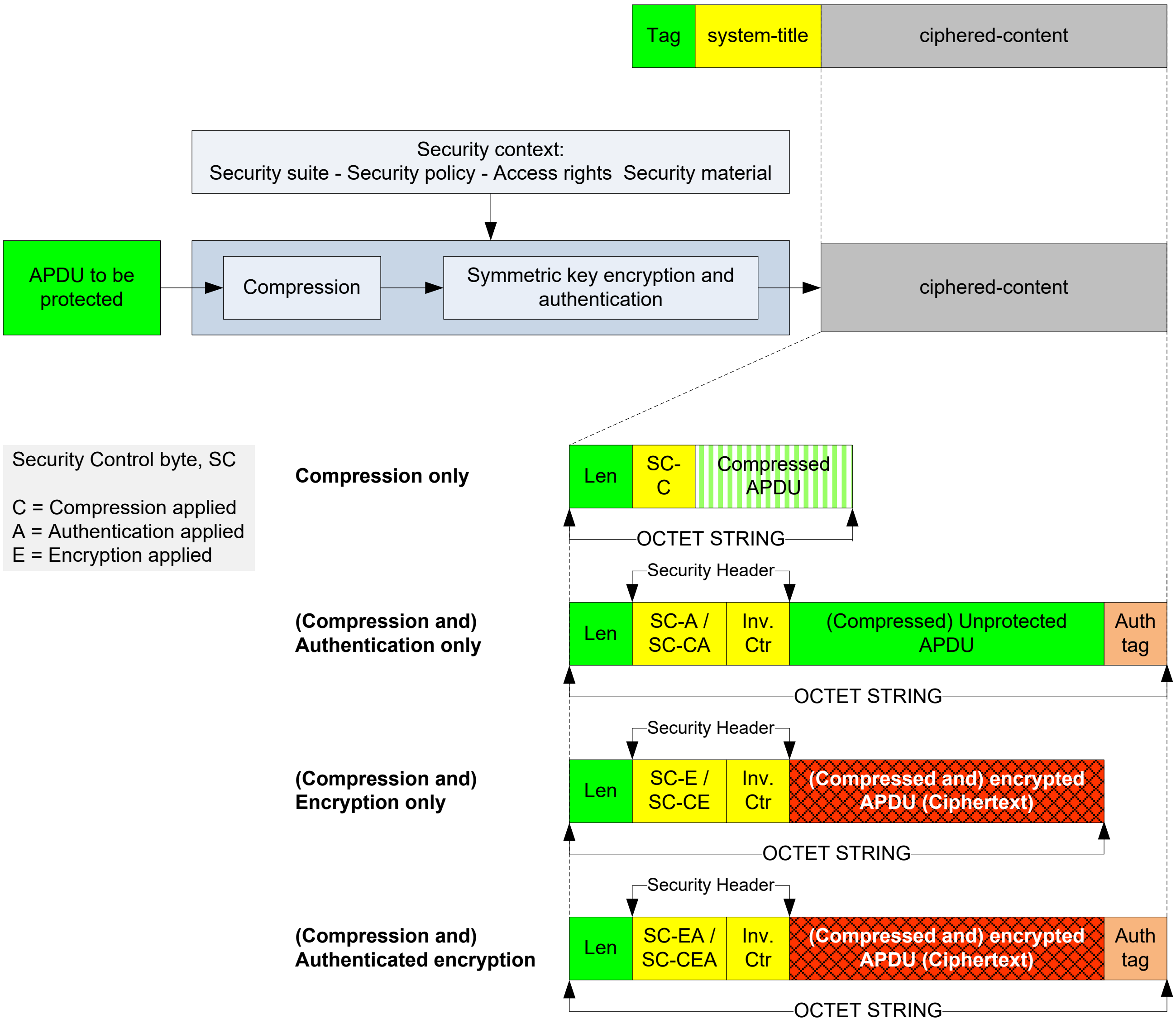

Tag
system-title
ciphered-content
Security context:
Security suite - Security policy - Access rights Security material
APDU to be protected
Compression
Symmetric key encryption and authentication
ciphered-content
Security Control byte, SC
C = Compression applied
A = Authentication applied
E = Encryption applied
Compression only
Len
SC-C
Compressed APDU
OCTET STRING
Security Header
(Compression and) Authentication only
Len
SC-A / SC-CA
Inv. Ctr
(Compressed) Unprotected APDU
Auth tag
OCTET STRING
Security Header
(Compression and) Encryption only
Len
SC-E / SC-CE
Inv. Ctr
(Compressed and) encrypted APDU (Ciphertext)
OCTET STRING
Security Header
(Compression and) Authenticated encryption
Len
SC-EA / SC-CEA
Inv. Ctr
(Compressed and) encrypted APDU (Ciphertext)
Auth tag
OCTET STRING

Figure 31 – Structure of general-glo-ciphering and general-ded-ciphering xDLMS APDUs

9.2.7.2.4.8 The general-ciphering APDU

The general-ciphering APDU can be used between a client and a server or between a third party and the server. These APDUs carry also the necessary information on the key to be used. The structure of the general-ciphering APDU is shown in Figure 32. See also the complete Green Book.

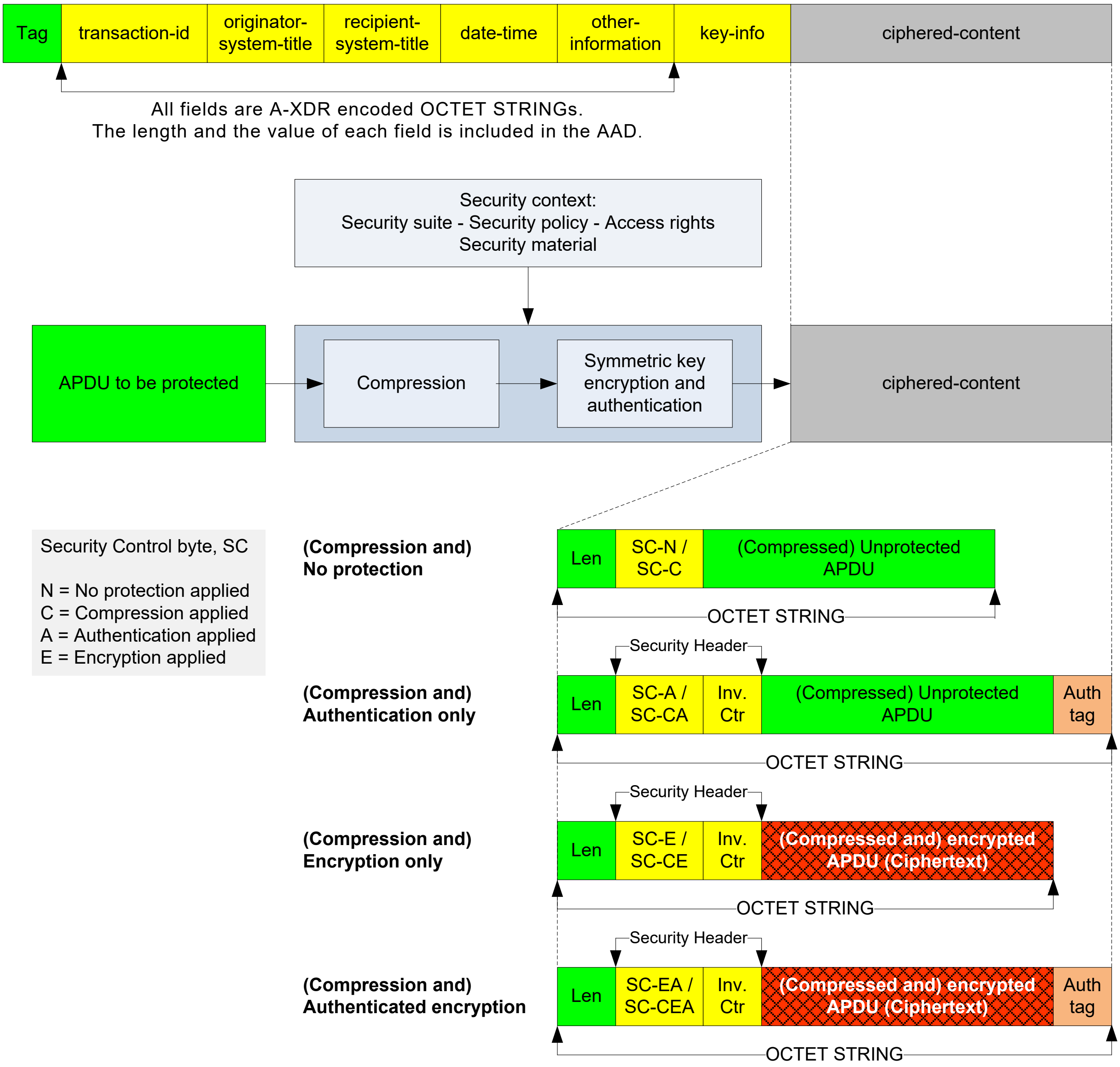

Tag
transaction-id
originator-system-title
recipient-system-title
date-time
other-information
key-info
ciphered-content
All fields are A-XDR encoded OCTET STRINGs.
The length and the value of each field is included in the AAD.
Security context:
Security suite - Security policy - Access rights
Security material
APDU to be protected
Compression
Symmetric key encryption and authentication
ciphered-content
Security Control byte, SC
N = No protection applied
C = Compression applied
A = Authentication applied
E = Encryption applied
(Compression and) No protection
Len
SC-N / SC-C
(Compressed) Unprotected APDU
OCTET STRING
Security Header
(Compression and) Authentication only
Len
SC-A / SC-CA
Inv. Ctr
(Compressed) Unprotected APDU
Auth tag
OCTET STRING
Security Header
(Compression and) Encryption only
Len
SC-E / SC-CE
Inv. Ctr
(Compressed and) encrypted APDU (Ciphertext)
OCTET STRING
Security Header
(Compression and) Authenticated encryption
Len
SC-EA / SC-CEA
Inv. Ctr
(Compressed and) encrypted APDU (Ciphertext)
Auth tag
OCTET STRING

Figure 32 – Structure of general-ciphering xDLMS APDUs

9.2.7.2.5 Digital signature

The algorithm is the elliptic curve digital signature algorithm (ECDSA) as specified in the complete Green Book.

The structure of the general-signing APDU is shown in Figure 33. For the additional fields, see the complete Green Book.

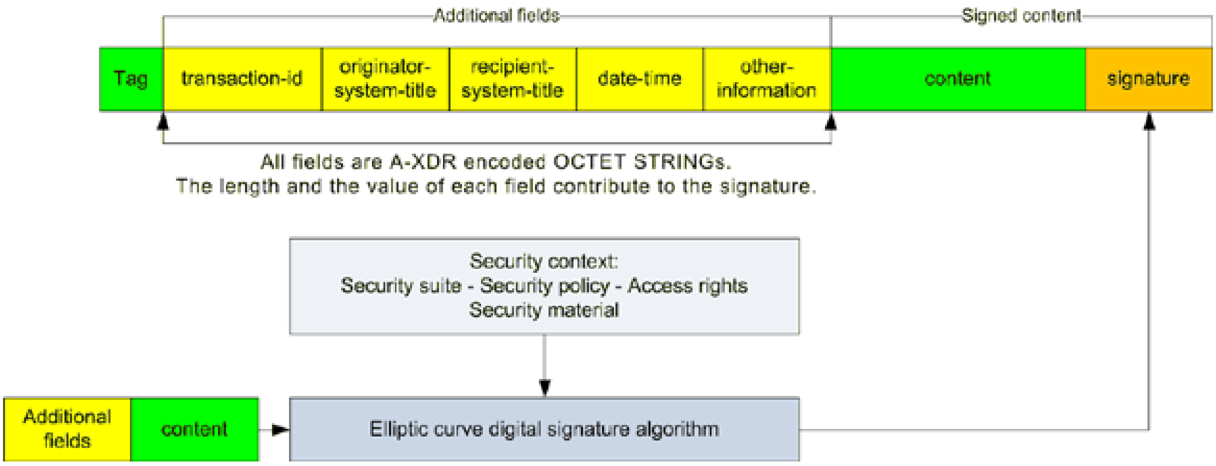

Additional fields
Signed content
Tag
transaction-id
originator-system-title
recipient-system-title
date-time
other-information
content
signature
All fields are A-XDR encoded OCTET STRINGs.
The length and the value of each field contribute to the signature.
Security context:
Security suite - Security policy - Access rights
Security material
Additional fields
content
Elliptic curve digital signature algorithm

Figure 33 – Structure of general-signing APDUs

For more information see the complete Green Book.

9.3 DLMS/COSEM application layer service specification

9.3.1 Service primitives and parameters

In general, the services of a layer (or sublayer) are the capabilities it offers to a user in the next higher layer (or sublayer). In order to provide its service, a layer builds its functions on the services it requires from the next lower layer. Figure 34 illustrates this notion of service hierarchy and shows the relationship of the two correspondent N-users and their associated N-layer peer protocol entities.

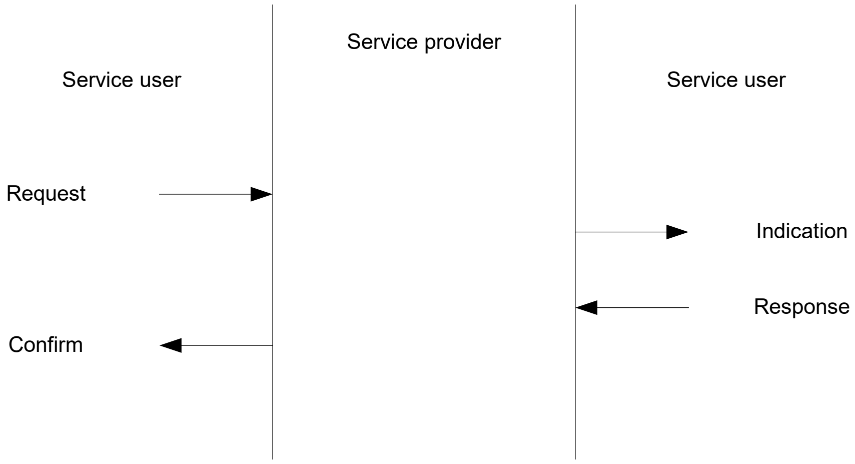


Figure 34 – Service primitives

Services are specified by describing the information flow between the N-user and the N-layer. This information flow is modelled by discrete, instantaneous events, which characterize the provision of a service. Each event consists of passing a service primitive from one layer to the other through an N-layer service access point associated with an N-user. Service primitives convey the information required in providing a particular service. These service primitives are an abstraction in that they specify only the service provided rather than the means by which the service is provided. This definition of service is independent of any particular interface implementation.

Services are specified by describing the service primitives and parameters that characterize each service. A service may have one or more related primitives that constitute the activity that is related to the particular service. Each service primitive may have zero or more parameters that convey the information required to provide the service. Primitives are of four generic types:

- REQUEST: The request primitive is passed from the N-user to the N-layer to request that a service be initiated;
- INDICATION: The indication primitive is passed from the N-layer to the N-user to indicate an internal N-layer event that is significant to the N-user. This event may be logically related to a remote service request, or may be caused by an event internal to the N-layer;
- RESPONSE: The response primitive is passed from the N-user to the N-layer to complete a procedure previously invoked by an indication primitive;
- CONFIRM: The confirm primitive is passed from the N-layer to the N-user to convey the results of one or more associated previous service request(s).

Possible relationships among primitive types are illustrated by the time-sequence diagrams shown in Figure 35. The figure also indicates the logical relationship of the primitive types. Primitive types that occur earlier in time and are connected by dotted lines in the diagrams are the logical antecedents of subsequent primitive types.

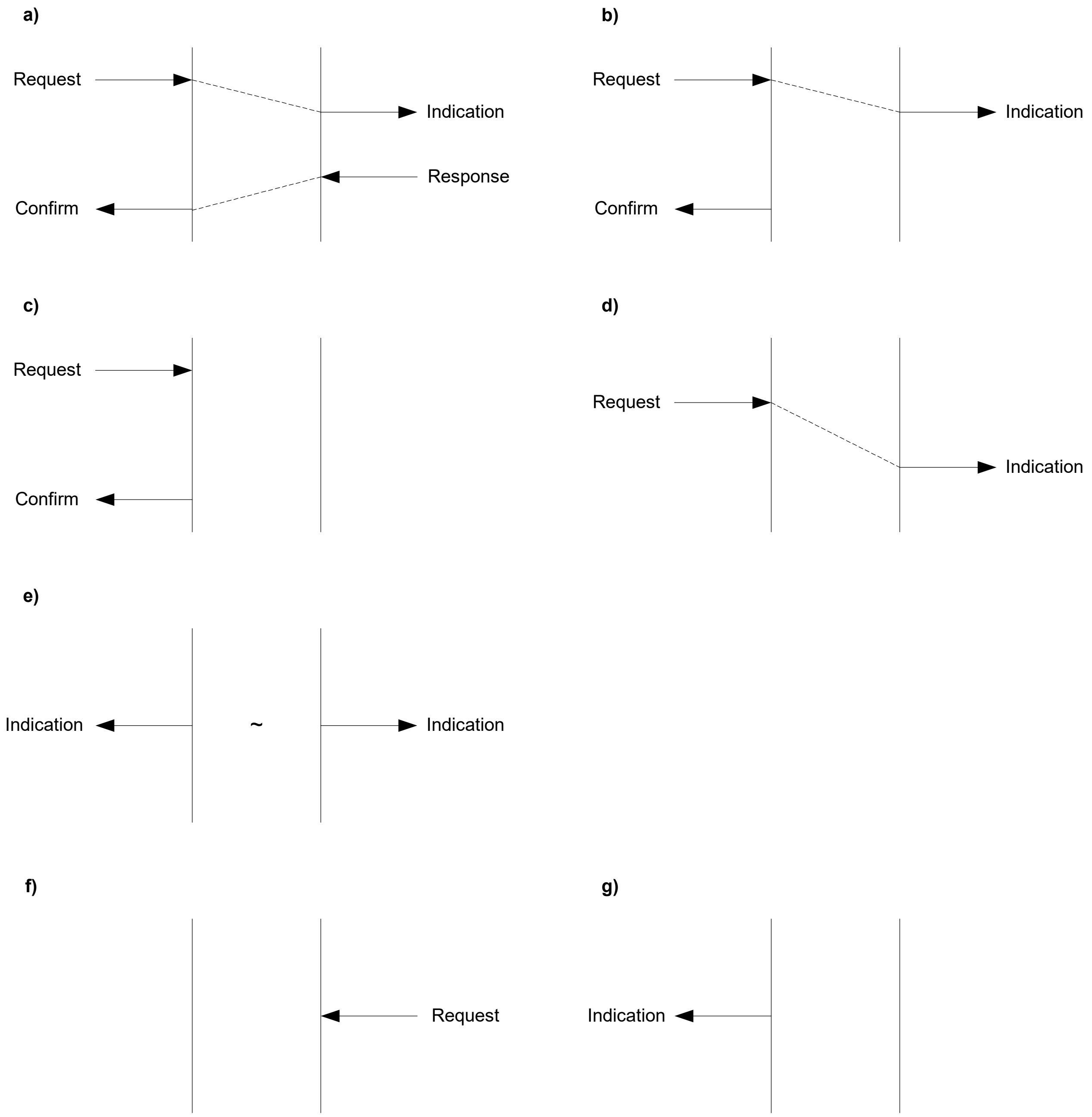


Figure 35 – Time sequence diagrams

The service parameters of the DLMS/COSEM AL service primitives are presented in a tabular format. Each table consists of two to five columns describing the service primitives and their parameters. In each table, one parameter – or a part of it – is listed on each line. In the appropriate service primitive columns, a code is used to specify the type of usage of the parameter. The codes used are listed in Table 10.

Some parameters may contain sub-parameters. These are indicated by labelling of the parameters as M, U, S or C, and indenting all sub-parameters under the parameter. Presence of the sub-parameters is always dependent on the presence of the parameter that they appear under. For example, an optional parameter may have sub-parameters; if the parameter is not supplied, then no sub-parameters may be supplied.

Table 10 – Codes for AL service parameters

M	The parameter is mandatory for the primitive.
U	The parameter is a user option, and may or may not be provided depending on dynamic usage by the ASE user.
S	The parameter is selected among other S-parameters as internal response of the server ASE environment.
C	The parameter is conditional upon other parameters or the environment of the ASE user.
(–)	The parameter is never present.
=	The " (=) " code following one of the M, U, S or C codes indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. For instance, an " M (=) " code in the .indication service primitive column and an "M" in the .request service primitive column means that the parameter in the .indication primitive is semantically equivalent to the one in the .request primitive.

Throughout this Technical Report, the following rules are observed regarding the naming of terms:

- the name of ACSE services and the data transfer services using LN referencing is written in uppercase. Examples are: COSEM-OPEN, GET;
- the name of the data transfer services using SN referencing is written in title case. Examples are: Read, Write;
- camel notation is used in the following cases: DataNotification, EventNotification, TriggerEventNotificationSending, UnconfirmedWrite, InformationReport;
- the types of the LN service primitives may be mentioned in two alternative forms. Examples: “GET.request service primitive of Request_Type == NORMAL” or “GET-REQUEST-NORMAL service primitive”;
- service parameter name elements are capitalized and joined with an underscore to signify a single entity: Examples are Protocol_Connection_Parameters and COSEM_Attribute_Descriptor;
- when the same parameter may occur several times, this is indicated by repeating the parameter in curly brackets. Example: Data { Data };
- in the data transfer service specifications, parameters used with block transfer only are shown in bold. Example: **DataBlock_G**;

  NOTE This applies only to the service-specific block transfer mechanism.

- direct reference to a service parameter uses the capitalized form, while indirect (non-specific) reference uses the small caps without underscore joining. A direct reference example is: “The COSEM_Attribute_Descriptor parameter references a COSEM object attribute.” An indirect (non-specific) reference example is: “A GET-REQUEST-NORMAL service primitive contains a single COSEM attribute descriptor”;
- the names of COSEM data transfer APDUs using LN referencing are capitalized and joined with a dash to signify a single entity. Example: Get-Request-Normal;
- the names of COSEM data transfer APDUs using SN referencing use the camel notation. Example: ReadRequest.

9.3.2 The COSEM-OPEN service

Function

The function of the COSEM-OPEN service is to establish an AA between peer COSEM Application Entities (AEs). It uses the A-ASSOCIATE service of the ACSE. The COSEM-OPEN service provides only the framework for *transporting* this information. To provide and verify that information is the job of the appropriate COSEM AP.

For more information see the complete Green Book.

9.3.3 The COSEM-RELEASE service

Function

The function of the COSEM-RELEASE service is to gracefully release an existing AA. Depending on the way it is invoked, it uses the A-RELEASE service of the ACSE or not.

For more information see the complete Green Book.

9.3.4 The COSEM-ABORT service

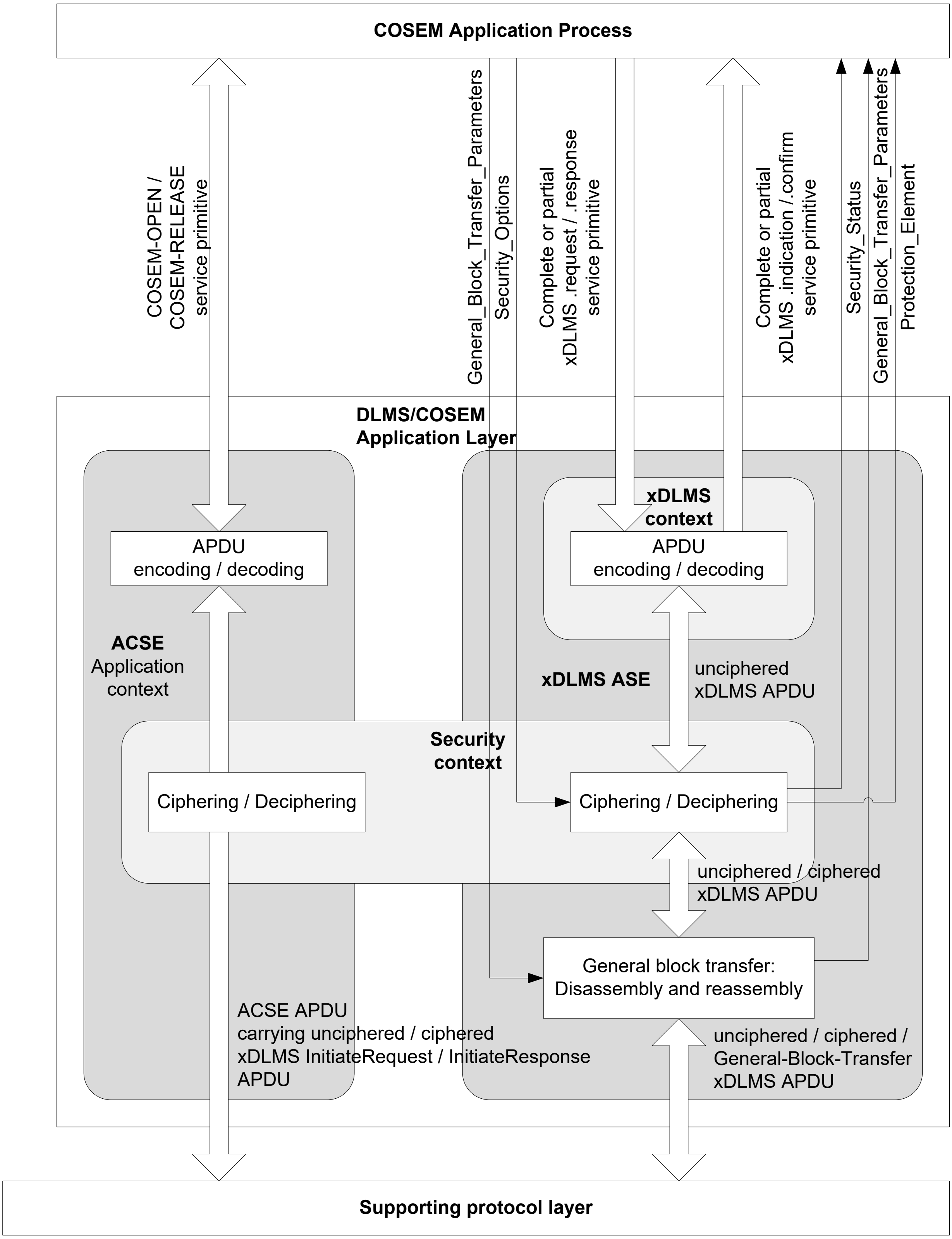
Function

The function of the COSEM-ABORT service is to indicate an unsolicited disconnection of the supporting layer.

For more information see the complete Green Book.

9.3.5 Protection and general block transfer parameters

To control cryptographic protection of xDLMS APDUs and the GBT mechanism, additional service parameters are passed between the AL and the AP as shown in Figure 36 and the complete Green Book.


COSEM Application Process
COSEM-OPEN / COSEM-RELEASE service primitive
General_Block_Transfer_Parameters
Security_Options
Complete or partial xDLMS .request / .response service primitive
Complete or partial xDLMS .indication /.confirm service primitive
Security_Status
General_Block_Transfer_Parameters
Protection_Element
DLMS/COSEM Application Layer
xDLMS context
APDU encoding / decoding
APDU encoding / decoding
ACSE Application context
xDLMS ASE
unciphered xDLMS APDU
Security context
Ciphering / Deciphering
Ciphering / Deciphering
unciphered / ciphered xDLMS APDU
General block transfer: Disassembly and reassembly
ACSE APDU carrying unciphered / ciphered xDLMS InitiateRequest / InitiateResponse APDU
unciphered / ciphered / General-Block-Transfer xDLMS APDU
Supporting protocol layer

NOTE For services initiated by the client, the service primitives are .request, .indication, .response and .confirm. For unsolicited services – initiated by the server – the service primitives are .request and .indication.

Figure 36 – Additional service parameters to control cryptographic protection and GBT

For more information see the complete Green Book.

9.3.6 The GET service

Function

The GET service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner. Its function is to read the value of one or more COSEM object attributes. The result can be delivered in a single response or – if it is too long to fit in a single response – in multiple responses, with block transfer.

For more information see the complete Green Book.

9.3.7 The SET service

Function

The SET service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner. Its function is to write the value of one or more COSEM object attributes. The data to be written can be sent in a single request or – if it is too long to fit in a single request – in multiple requests, with block transfer.

For more information see the complete Green Book.

9.3.8 The ACTION service

Function

The ACTION service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner. Its function is to invoke one or more COSEM objects methods. It comprises two phases:

- in the first phase, the client sends the reference(s) of the method(s) to be invoked, with the method invocation parameters necessary;
- in the second phase, after invoking the methods, the server sends back the result and the return parameters generated by the invocation of the method(s), if any.

If the method invocation parameters are too long to fit in a single request, they are sent in multiple requests (block transfer from the client to the server). If the result and the return parameters are too long to fit in a single response, they are returned in multiple responses (block transfer from the server to the client.)

For more information see the complete Green Book.

9.3.9 The ACCESS service

9.3.9.1 Overview – Main features

9.3.9.1.1 General

The ACCESS service is a unified service which can be used to access multiple COSEM object attributes and/or methods with a single .request / .response. The purpose of introducing it is to improve xDLMS messaging while maintaining co-existence with the existing xDLMS services.

9.3.9.1.2 Unified WITH-LIST service to improve efficiency

The ACCESS service is a unified service using LN referencing that can be used to read or write multiple COSEM object attributes and/or to invoke multiple methods with a single .request / .response. Each request contains a list of requests and related data. Each response contains a list of return data and the result of the request.

NOTE SN referencing is currently not supported. It can be added by introducing new variants of the service.

Whereas GET- / SET- / ACTION-WITH-LIST service requests can include one request type –GET, SET and/or ACTION – only on the list, ACCESS service requests can include different request types. This allows reducing the number of exchanges and thereby improves efficiency.

The processing of the list of requests starts at the first request on the list and continues with processing the next one until the end is reached.

9.3.9.1.3 Specific variants for selective access

The GET / SET .request service primitives shall always contain Access_Selection_Parameters even in the case when selective access is not available or not needed. In contrast, the ACCESS service provides specific variants to access attributes without or with selective access. This obviates the need to include Access_Selection_Parameters when selective access is not available or not needed thereby reducing overhead and improving efficiency.

9.3.9.1.4 Long_Invoke_Id parameter

The Invoke-Id parameter of the GET, SET and ACTION services allows the client and the server to pair requests and responses. The range of the Invoke_Id is 0…15.

In some cases this is not sufficient. To support those cases, the ACCESS service uses a Long_Invoke_Id parameter. The range of the Long_Invoke_Id is 0…16 777 215.

NOTE Description of the circumstances when long Invoke_id-s are useful is beyond the Scope of this Technical Report.

9.3.9.1.5 Self-descriptive responses

When requested by the client, the ACCESS.response service primitive carries not only the response to each request, i.e. the result of accessing each attribute / method and the return data but also the Access_Request_Specification service parameter – carrying the attribute / method references and where applicable, the Access_Selection parameters – rendering the .response service primitive self-descriptive. Such self-descriptive responses can be stored and processed on their own, without the need to pair responses and requests.

9.3.9.1.6 Failure management

In the case of the GET- / SET- / ACTION-WITH-LIST services the client cannot control what should happen if one of the requests fails. In contrast, the ACCESS service allows the client to control if the requests that follow the failed one on the list should be processed or not.

9.3.9.1.7 Time stamp as a service parameter

The xDLMS services specified earlier do not provide a service parameter in the .request or in the .response service primitive to carry a time stamp.

In contrast, ACCESS service primitives provide a service parameter to carry the time stamp holding the date and time of invoking the service primitive. This further reduces overhead.

9.3.9.1.8 Presence of data in service primitives

There are important differences between the GET / SET / ACTION services and the ACCESS service as regards to data in the service primitives:

- GET service: data is not present in the request. In the response, either data or result (Data-Access-Result) is returned;
- SET service: data is present in the request. In the response only result (Data-Access-Result) is returned;
- ACTION service: method invocation parameters are optional in the request. In the response the result of invoking the method (Action-Result) and optionally the result of returning the return parameters (Data or Data-Access-Result) is returned;

- ACCESS service: data is associated with each attribute / method reference in the request. If data is not needed for a particular request, then null-data is included. In the response, both data and result are returned. If there is no data to return for a particular response, then null-data is included. In the case of accessing a method, Access-Response-Action (Action-Result) conveys both the result of invoking the method and the result of returning the return parameters.

9.3.9.2 Service specification

Function

The ACCESS service is a unified service using LN referencing that can be used to read or write multiple COSEM object attributes and/or to invoke multiple methods with a single .request / .response. Each request contains a list of requests and related data. Each response contains a list of return data and the result of the request. It can be invoked in a confirmed or unconfirmed manner. It can be used with the general block transfer and general ciphering mechanisms.

The use of the conformance block is the following:

- bit 17 *access* indicates the support of the ACCESS service;
- bit 1 *general-protection* indicates the availability of the general protection APDUs;
- bit 2 *general-block-transfer* indicates the availability of the GBT mechanism;
- bit 8 *attribute0-supported-with-set* and bit 10 *attribute0-supported-with-get* (10) are not relevant: attribute0 is always supported;
- bit 9 priority-mgmt-supported is relevant;
- bit 14 *multiple-references* is irrelevant: the ACCESS service always supports multiple references;
- bit 21 *selective-access* is relevant. The access selection parameters can be used only if the use of selective access has been successfully negotiated.

For more information see the complete Green Book.

9.3.10 The DataNotification service

Function

The DataNotification service is an unsolicited, unconfirmed service. It is used by the server to push data to the client. It is an unconfirmed service. The push process is configured by “Push setup” objects; see DLMS UA 1000-1 Ed. 13:2019, 4.4.8.

For more information see the complete Green Book.

9.3.11 The EventNotification service

Function

The EventNotification service is an unsolicited service initiated by the server upon the occurrence of an event in order to inform the client of the value of an attribute as though it had been requested by the COSEM. It is an unconfirmed service.

For more information see the complete Green Book.

9.3.12 The TriggerEventNotificationSending service

Function

The function of the TriggerEventNotificationSending service is to trigger the server by the client to send the frame carrying the EventNotification.request APDU.

This service is necessary in the case of protocols, when the server is not able to send a real non-solicited EventNotification.request APDU.

For more information see the complete Green Book.

9.3.13 Variable access specification

Variable_Access_Specification is a parameter of the xDLMS Read / Write / UnconfirmedWrite InformationReport .request / .indication service primitives. Its variants are shown in the complete Green Book:

- Variable_Name identifies a DLMS named variable;
- Parameterized_Access provides the capability to transport additional parameters;
- Block_Number_Access transports a block number;
- Read_Data_Block_Access transports block transfer control information and raw data;
- Write_Data_Block_Access transports block transfer control information.

The use of the different variants depends on the service and it is described in the respective SN service specifications.

For more information see the complete Green Book.

9.3.14 The Read service

Function

The Read service is used with SN referencing. It is a confirmed service. Its functions are:

- to read the value of one or more COSEM object attributes. In this case, the encoded form of the .request service primitive shall fit in a single APDU. The result can be delivered in a single response, or – if it is too long to fit in a single response – in multiple responses, with block transfer;
- to invoke one or more COSEM object methods, when return parameters are expected. In this case, if either the .request (including the method references and the method invocation parameters) or the .response service primitive (including the results and return parameters) is too long to fit in a single APDU, then block transfer with multiple requests and/or responses can be used.

The Read service is specified in IEC 61334-4-41:1996, 10.4 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification is reproduced here, together with the extensions made for DLMS/COSEM.

For more information see the complete Green Book.

9.3.15 The Write service

Function

The Write service is used with SN referencing. It is a confirmed service. Its functions are:

- to write the value of one or more COSEM object attributes;
- to invoke one or more COSEM object methods when no return parameters are expected.

In both cases, if the encoded form of the .request service primitive does not fit in a single APDU, then it can be sent in several requests with block transfer. The .response service primitive shall always fit in a single APDU.

The Write service is specified in IEC 61334-4-41:1996, 10.5 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification is reproduced here, together with the extensions made for DLMS/COSEM.

For more information see the complete Green Book.

9.3.16 The UnconfirmedWrite service

Function

The UnconfirmedWrite service is used with SN referencing. It is an unconfirmed service. Its functions are:

- to write the value of one or more COSEM object attributes;
- to invoke one or more COSEM object method when no return parameters are expected.

The UnconfirmedWrite.request service primitive shall always fit in a single APDU.

The UnconfirmedWrite service is specified in IEC 61334-4-41:1996, 10.6 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification is reproduced here, together with the extensions made for DLMS/COSEM.

For more information see the complete Green Book.

9.3.17 The InformationReport service

Function

The InformationReport service is an unsolicited service initiated by the server upon the occurrence of an event in order to inform the client of the value of one or more DLMS named variables – mapped to COSEM object attributes – as though they had been requested by the client. It is an unconfirmed service.

The InformationReport service is specified in IEC 61334-4-41:1996, 10.7 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification of the InformationReport service is reproduced here, together with the extensions made for DLMS/COSEM.

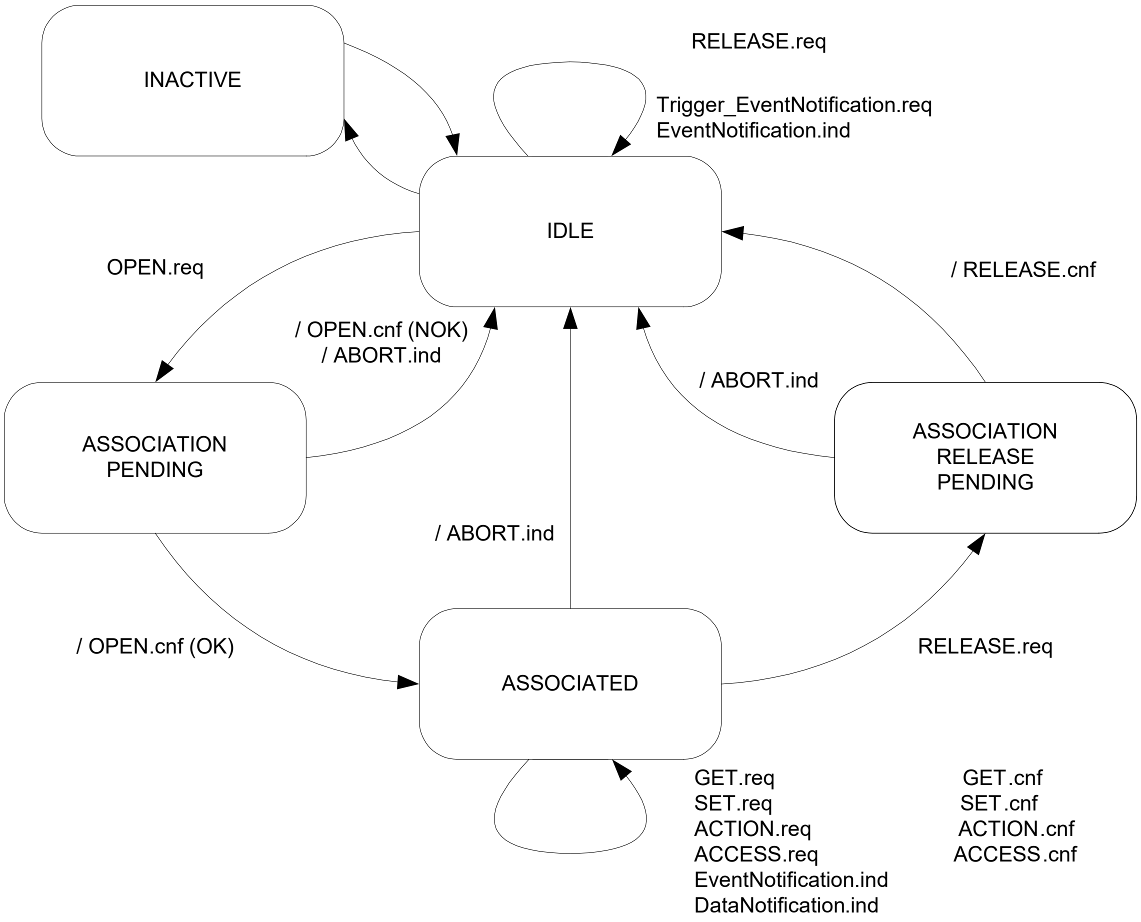
For more information see the complete Green Book.

9.4 DLMS/COSEM application layer protocol specification

9.4.1 The control function (CF)

9.4.1.1 State definitions of the client side control function

Figure 37 shows the state machine for the client side CF, see also Figure 25.



NOTE 1 On the state diagrams of the client and server CF, the following conventions are used:

- service primitives with no "/" character as first character are "stimulants": the invocation of these primitives is the origin of the state transition;
- service primitives with an "/" character as first character are "outputs": the generation of these primitives is done on the state transition path.

Figure 37 – Partial state machine for the client side control function

The state definitions of the client CF – and of the AL including the CF – are as follows:

INACTIVE In this state, the CF has no activity at all: it neither provides services to the AP nor uses services of the supporting protocol layer.

IDLE This is the state of the CF when there is no AA existing, being released, or being established [3]. Nevertheless, some data exchange between the client and server – if the physical channel is already established – is possible. The CF can handle the EventNotification service.

NOTE 2 State transitions between the INACTIVE and IDLE states are controlled outside of the protocol. For example, it can be considered that the CF makes the state transition from INACTIVE to IDLE by being instantiated and bound on the top of the supporting protocol layer. The opposite transition may happen by deleting the given instance of the CF.

[3] Note, that it is the state machine for the AL: lower layer connections, including the physical connection, are not taken into account. On the other hand, physical connection establishment is done outside of the protocol.

ASSOCIATION PENDING The CF leaves the IDLE state and enters this state when the AP requests the establishment of an AA by invoking the COSEM-OPEN.request primitive (OPEN.req). The CF may exit this state and enter either the ASSOCIATED state or return to the IDLE state, and generates the COSEM-OPEN.confirm primitive, (/OPEN.cnf(OK)) or (/OPEN.cnf(NOK)), depending on the result of the association request. The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

ASSOCIATED The CF enters this state when the AA has been successfully established. All xDLMS services and APDUs are available in this state. The CF remains in this state until the AP requests the release of the AA by invoking the COSEM-RELEASE.request primitive (RELEASE.req). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

ASSOCIATION RELEASE PENDING The CF leaves the ASSOCIATED state and enters this state when the AP requests the release of the AA by invoking the COSEM-RELEASE.request primitive (RELEASE.req). The CF remains in this state, waiting for the response to this request from the server. As the server is not allowed to refuse a release request, after exiting this state, the CF always enters the IDLE state. The CF may exit this state by generating the COSEM-RELEASE.confirm primitive following the reception of a response form the server or by generating it locally (/RELEASE.cnf). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

9.4.1.2 State definitions of the server side control function

Figure 38 shows the state machine for the server side CF, see Figure 25.

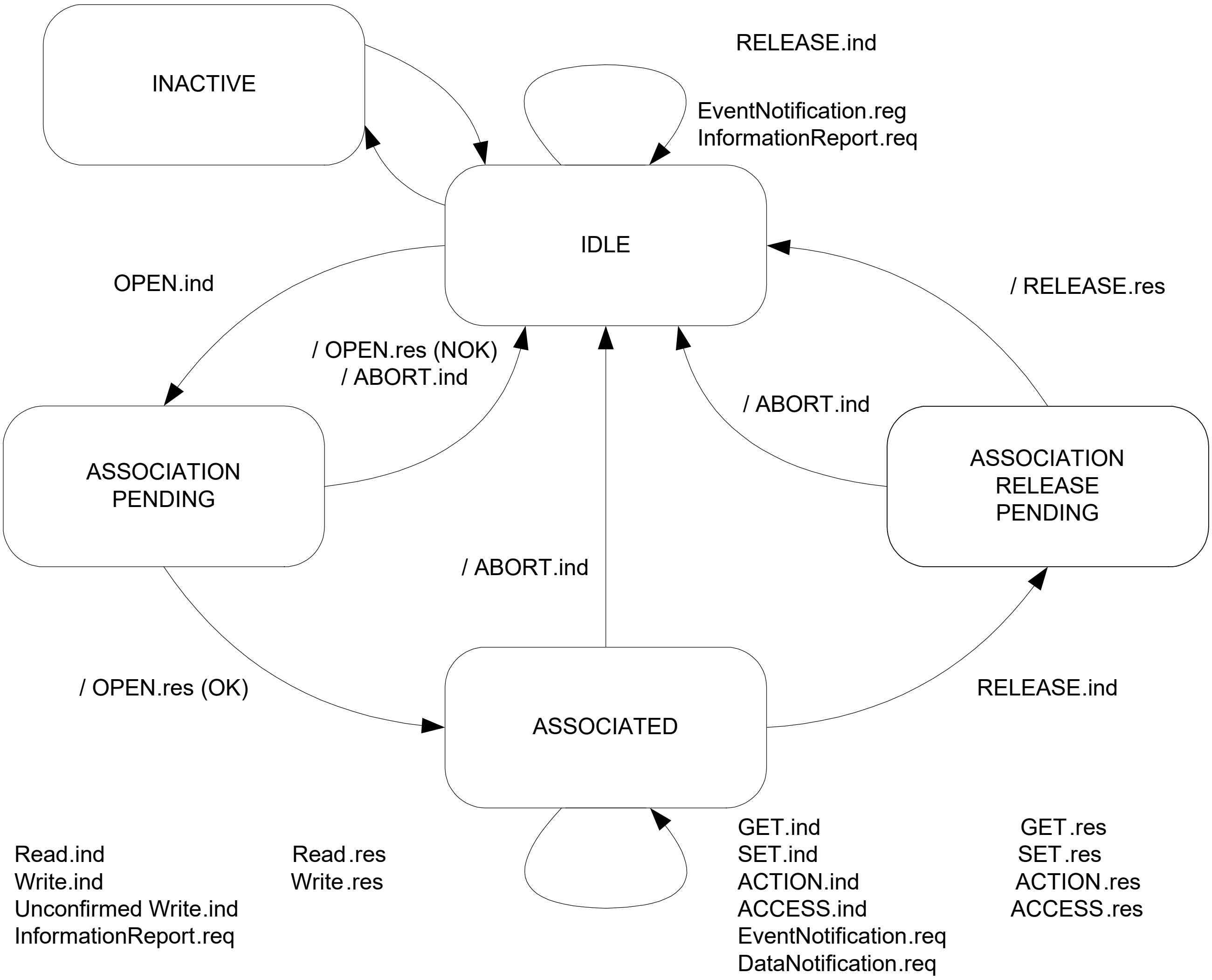


Figure 38 – Partial state machine for the server side control function

INACTIVE — In this state, the CF has no activity at all: it neither provides services to the AP nor uses services of the supporting protocol layer.

IDLE — This is the state of the CF when there is no AA existing, being released, or being established [4]. Nevertheless, some data exchange between the client and server – if the physical channel is already established – is possible. The CF can handle the EventNotification / InformationReport services.

ASSOCIATION PENDING — The CF leaves the IDLE state and enters this state when the client requests the establishment of an AA, and the server AL generates the COSEM-OPEN.indication primitive (/OPEN.ind). The CF may exit this state and enter either the ASSOCIATED state or return to the IDLE state, depending on the result of the association request, and invokes the COSEM-OPEN.response primitive, (/OPEN.res(OK)) or (/OPEN.res(NOK)). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

[4] Note that it is the state machine for the AL: lower layer connections, including the physical connection, are not taken into account. On the other hand, physical connection establishment is done outside of the protocol.

ASSOCIATED	The CF enters this state when the AA has been successfully established. All xDLMS services and APDUs are available in this state. The CF remains in this state until the client requests the release of the AA, and the server AL generates the COSEM-RELEASE.ind primitive (/RELEASE.ind). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).
ASSOCIATION RELEASE PENDING	The CF leaves the ASSOCIATED state and enters this state when the client request the release of an AA, and the server AP receives the COSEM-RELEASE.indication primitive (/RELEASE.ind). The CF remains in this state, waiting that the AP accepts the release request. As the server is not allowed to refuse a release request, after exiting this state, the CF always enters the IDLE state. The CF may exit this state when the AP accepts the release of the AA, and invokes the COSEM-RELEASE.response primitive (RELEASE.res). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

9.4.2 The ACSE services and APDUs

9.4.2.1 ACSE functional units, services and service parameters

The DLMS/COSEM AL ACSE is based on the connection-oriented ACSE, as specified in ISO/IEC 15953:1999 and ISO/IEC 15954:1999.

Functional units are used to negotiate ACSE user requirements during association establishment. Five functional units are defined:

- Kernel functional unit;
- Authentication functional unit;
- ASO-context negotiation functional unit;

NOTE 1 ISO/IEC 15953:1999 and ISO/IEC 15954:1999 use the term ‘ASO-context”. In DLMS/COSEM the term ‘Application context” is used as in ISO/IEC 8649 / ISO/IEC 8650.

- Higher Level Association functional unit; and
- Nested Association functional unit.

The DLMS/COSEM AL uses only the Kernel and the Authentication functional unit.

The acse-requirements parameters of the AARQ and AARE APDUs are used to select the functional units for the association.

The Kernel functional unit is always available and includes the basic services A-ASSOCIATE, A-RELEASE.

The Authentication functional unit supports authentication during association establishment. The availability of this functional unit is negotiated during association establishment. This functional unit does not include additional services. It adds parameters to the A-ASSOCIATE service.

Table 11 shows the services, APDUs and APDU fields associated with the ACSE functional units, as used by the DLMS/COSEM AL. The abstract syntax of the ACSE APDUs is specified in 9.5.

Table 11 – Functional Unit APDUs and their fields

Functional unit	Service	APDU	Field name	Presence
Kernel	A-ASSOCIATE	AARQ	protocol-version	O
			application-context-name	M
			called-AP-title	U
			called-AE-qualifier	U
			called-AP-invocation-identifier	U
			called-AE-invocation-identifier	U
			calling-AP-title	U
			calling-AE-qualifier	U
			calling-AP-invocation-identifier	U
			calling-AE-invocation-identifier	U
			implementation-information	O
			user-information [2)]	M
			(carrying an xDLMS Initiate.request APDU)	
			dedicated-key	U
			response-allowed	U
			proposed-quality-of-service	U
			proposed-dlms-version-number	M
			proposed-conformance	M
			client-max-receive-pdu-size	M
		AARE	protocol-version	O
			application-context-name	M
			result	M
			result-source-diagnostic	M
			responding-AP-title	U
			responding-AE-qualifier	U
			responding-AP-invocation-identifier	U
			responding-AE-invocation-identifier	U
			implementation-information	O
			user-information [3)]	M
			(carrying an xDLMS initiateResponse APDU)	S
			negotiated-quality-of-service	U
			negotiated-dlms-version-number	M
			negotiated-conformance	M
			server-max-receive-pdu-size	M
			vaa-name	M
			(or carrying a confirmedServiceError APDU)	S
	A-RELEASE	RLRQ	reason	U
			user-information	U
		RLRE	reason	U
			user-information	U
Authentication	A-ASSOCIATE	AARQ	sender-acse-requirements	U
			mechanism-name	U
			calling-authentication-value	U
		AARE	responder-acse-requirements	U
			mechanism-name	U
			responding-authentication-value	U

NOTE 1 This table is based on ISO/IEC 15954:1999, Table 2 and 3. The fields are listed in the order as they are in the ACSE APDUs.

M Presence is mandatory
O Presence is ACPM option
U Presence is ACSE service-user option
S The parameter is selected among other S-parameters as internal response of the server ASE environment.

NOTE 2 According to ISO/IEC 15953:1999 the user-information parameter is optional. However, in the DLMS/COSEM environment it is mandatory in the AARQ / AARE APDUs.

There are several changes in ISO/IEC 15953:1999 and ISO/IEC 15954:1999 compared to ISO/IEC 8649 and ISO/IEC 8650-1:

- In ISO/IEC 15954, protocol-version is mandatory in the AARQ and optional in the AARE. In DLMS/COSEM it is kept as mandatory for backward compatibility;
- Instead of “application-context-name”, “ASO-context-name” is used. In DLMS/COSEM, “application-context-name” is kept. ISO/IEC15954 7.1.5.2 specifies this: the ASO-context-name is optional. If backward compatibility with older implementations of ACSE is desired, it must be present. Therefore, in DLMS/COSEM it is mandatory;
- In ISO/IEC 15954, the result and result-source-diagnostic parameters are optional. ISO/IEC 15954 7.1.5.8 and 7.1.5.9 specifies this: The Result / Result-source-diagnostic are optional. If backward compatibility with older implementations of ACSE is desired, it must be present. Therefore, in DLMS/COSEM these parameters are mandatory.

In general, the value of each field of the AARQ APDU is determined by the parameters of the COSEM-OPEN.request service primitive. Similarly, the value if each field of the AARE is determined by the COSEM-OPEN.response primitive. The COSEM-OPEN service is specified in 9.3.2.

The fields of the AARQ and AARE APDU are specified below. Managing these fields is specified in 9.4.4.1.

- protocol-version: the DLMS/COSEM AL uses the default value version 1. For details see ISO/IEC 15954:1999;
- application-context-name: COSEM application context names are specified in 9.4.2.2.2;

  NOTE 2 ISO/IEC 15953:1999 and ISO/IEC 15954:1999 use “ASO-context-name”
- called-, calling- and responding- titles, qualifiers and invocation-identifiers: these optional fields carry the value of the respective parameters of the COSEM-OPEN service. For details see ISO/IEC 15954:1999;
- implementation-information: this field is not used by the DLMS/COSEM AL. For details see ISO/IEC 15954:1999;
- user-information: in the AARQ APDU, it carries an xDLMS InitiateRequest APDU holding the elements of the Proposed_xDLMS_Context parameter of the COSEM-OPEN.request service primitive. In the AARE APDU, it carries an xDLMS InitiateResponse APDU, holding the elements of the Negotiated_xDLMS_Context parameter, or an xDLMS confirmedServiceError APDU, holding the elements of the xDLMS_Initiate_Error parameter of the COSEM-OPEN.response service primitive;
- sender- and responder-acse-requirements: this field is used to select the optional functional units of the AARQ / AARE. In COSEM, only the Authentication functional unit is used. When present, it carries the value of BIT STRING { authentication (0) }. Bit set: authentication functional unit selected;
- mechanism-name: COSEM authentication mechanism names are specified in 9.4.2.2.3;
- calling- and responding- authentication-value: see 9.2.2.2.2;
- result: the value of this field is determined by the COSEM AP (acceptor) or the DLMS/COSEM AL (ACPM) as specified below. It is used to determine the value of the Result parameter of the COSEM-OPEN.confirm primitive:
  - if the AARQ APDU is rejected by the ACPM (i.e. the COSEM-OPEN.indication primitive is not issued by the DLMS/COSEM AL), the value “rejected (permanent)” or “rejected (transient)” is assigned by the ACPM;
  - otherwise, the value is determined by the Result parameter of the COSEM-OPEN.response APDU;

- result-source-diagnostic: this field contains both the Result source value and the Diagnostic value. It is used to determine the value of the Failure_Type parameter of the COSEM-OPEN.confirm primitive:
  - Result-source value: if the AARQ is rejected by the ACPM, (i.e. the COSEM-OPEN.indication primitive is not issued by the DLMS/COSEM AL) the ACPM assigns the value "ACSE service-provider". Otherwise, the ACPM assigns the value "ACSE service-user";
  - Diagnostic value: If the AARQ is rejected by the ACPM, the appropriate value is assigned by the ACPM. Otherwise, the value is determined by the Failure_Type parameter of the COSEM-OPEN.response primitive. If the Diagnostic parameter is not included in the .response primitive, the ACPM assigns the value "null".

The parameters of the RLRQ / RLRE APDUs – used when the COSEM-RELEASE service (see 9.3.3) is invoked with the parameter Use_RLRQ_RLRE == TRUE – are specified below.

- reason: carries the appropriate value as specified in 9.3.2;
- user-information: if present, it carries an xDLMS InitiateRequest / InitiateResponse APDU, holding the elements of the Proposed_xDLMS_Context / Negotiated_xDLMS_Context parameter of the COSEM-RELEASE.request / .response service primitive respectively. See 9.3.2.

9.4.2.2 Registered COSEM names

9.4.2.2.1 General

Within an OSI environment, many different types of network objects must be identified with globally unambiguous names. These network objects include abstract syntaxes, transfer syntaxes, application contexts, authentication mechanism names, etc. Names for these objects in most cases are assigned by the committee developing the particular basic ISO standard or by implementers' workshops, and should be registered. For DLMS/COSEM, these object names are assigned by the DLMS UA, and are specified below.

The decision no. 1999.01846 of OFCOM, Switzerland, attributes the following prefix for object identifiers specified by the DLMS User Association.

{ joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) }
NOTE As specified in ITU-T X.660 A.2.4, for historical reasons, the secondary identifiers ccitt and joint-iso-ccitt are synonyms for itu-t and joint-iso-itu-t, respectively, and thus may appear in ASN.1 OBJECT IDENTIFIER values, and also identify the corresponding primary integer value.

For DLMS/COSEM, object identifiers are specified for naming the following items:

- COSEM application context names;
- COSEM authentication mechanism names;
- cryptographic algorithm ID-s.

9.4.2.2.2 The COSEM application context

In order to effectively exchange information within an AA, the pair of AE-invocations shall be mutually aware of, and follow a common set of rules that govern the exchange. This common set of rules is called the application context of the AA. The application context that applies to an AA is determined during its establishment.ó

An AA has only one application context. However, the set of rules that make up the application context of an AA may contain rules for alteration of that set of rules during the lifetime of the AA.

The following methods may be used:

- identifying a pre-existing application context definition;
- transferring an actual description of the application context.

In the COSEM environment, it is intended that an application context pre-exists and it is referenced by its name during the establishment of an AA. The application context name is specified as OBJECT IDENTIFIER ASN.1 type. COSEM identifies the application context name by the following object identifier value:

```
COSEM_Application_Context_Name ::=
{joint-iso-ccitt(2)  country(16)  country-name(756)  identified-organization(5)  DLMS-UA(8)  application-context(1)
context_id(x)}
```

The meaning of this general COSEM application context is:

- there are two ASEs present within the AE invocation, the ACSE and the xDLMS ASE;
- the xDLMS ASE is as it is specified in IEC 61334-4-41:1996;

NOTE With the COSEM extensions to DLMS, see 9.1.4.

- the transfer syntax is A-XDR.

The specific context_id-s and the use of ciphered and unciphered APDUs are shown in Table 12:

Table 12 – COSEM application context names

Application context name	context_id	Unciphered APDUs	Ciphered APDUs
Logical_Name_Referencing_No_Ciphering ::=	context_id(1)	Yes	No
Short_Name_Referencing_No_Ciphering ::=	context_id(2)	Yes	No
Logical_Name_Referencing_With_Ciphering ::=	context_id(3)	Yes	Yes
Short_Name_Referencing_With_Ciphering ::=	context_id(4)	Yes	Yes

In order to successfully establish an AA, the application-context-name parameter of the AARQ and AARE APDUs should carry one of the “valid” names. The client proposes an application context name using the Application_Context_Name parameter of the COSEM-OPEN.request primitive. The server may return any value; either the value proposed or the value it supports.

9.4.2.2.3 The COSEM authentication mechanism name

Authentication of the client, the server or both is one of the security aspects addressed by the DLMS/COSEM specification. Three authentication security levels are specified:

- no security (Lowest Level Security) authentication, see 9.2.2.2.2.2;
- Low Level Security (LLS) authentication, see 9.2.2.2.2.3;
- High Level Security (HLS) authentication, see 9.2.2.2.2.4.

DLMS/COSEM identifies the authentication mechanisms by the following general object identifier value:

```
COSEM_Authentication_Mechanism_Name ::=
{joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8)
authentication_mechanism_name(2) mechanism_id(x)}
```

The value of the mechanism_id element selects one of the security mechanisms specified:

Table 13 – COSEM authentication mechanism names

COSEM_lowest_level_security_mechanism_name ::=	mechanism_id(0)
COSEM_low_level_security_mechanism_name ::=	mechanism_id(1)
COSEM_high_level_security_mechanism_name ::=	mechanism_id(2)
COSEM_high_level_security_mechanism_name_using_MD5 ::=	mechanism_id(3)
COSEM_high_level_security_mechanism_name_using_SHA-1 ::=	mechanism_id(4)
COSEM_high_level_security_mechanism_name_using_GMAC ::=	mechanism_id(5)
COSEM_high_level_security_mechanism_name_using_SHA-256 ::=	mechanism_id(6)
COSEM_high_level_security_mechanism_name_using_ECDSA ::=	mechanism_id(7)
NOTE 1 With mechanism_id(2), the method of processing the challenge is secret. NOTE 2 The use of authentication mechanisms 3 and 4 are not recommended for new implementations.	

When the Authentication_Mechanism_Name is present in the COSEM-OPEN service, the authentication functional unit of the A-ASSOCIATE service shall be selected. The process of LLS and HLS authentication is described in 9.2.2.2.2 and in the complete Green Book.

9.4.2.2.4 Cryptographic algorithm ID-s

Cryptographic algorithm IDs identify the algorithm for which a derived secret symmetrical key will be used. See the complete Green Book.

Cryptographic algorithms are identified by the following general object identifier value:

```
COSEM_Cryptographic_Algorithm_Id ::=
{joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) cryptographic-algorithms (3)
algorithm_id(x)}
```

The values of the algorithm_id-s are shown in Table 14. See also the complete Green Book.

Table 14 – Cryptographic algorithm ID-s

COSEM_cryptographic_algorithm_name_aes-gcm-128 ::=	algorithm_id(0)
COSEM_cryptographic_algorithm_name_aes-gcm-256 ::=	algorithm_id(1)
COSEM_cryptographic_algorithm_name_aes-wrap-128 ::=	algorithm_id(2)
COSEM_cryptographic_algorithm_name_aes-wrap-256 ::=	algorithm_id(3)

9.4.3 APDU encoding rules

9.4.3.1 Encoding of the ACSE APDUs

The ACSE APDUs shall be encoded in BER (ISO/IEC 8825-1:2015). The user-information parameter of these APDUs shall carry the xDLMS InitiateRequest / InitiateResponse / confirmedServiceError APDU as appropriate, encoded in A-XDR, and then encoding the resulting OCTET STRING in BER.

Examples for AARQ/AARE APDU encoding are given in Clauses 11 and 12.

9.4.3.2 Encoding of the xDLMS APDUs

The xDLMS APDUs shall be encoded in A-XDR, as specified in IEC 61334-6:2000.

9.4.3.3 XML

Depending on the parametrization of the “Push setup” object the DataNotification APDU can be encoded as an XML document using the XML schema specified in 9.6.

NOTE The use of XML to encode the other APDUs is not in the Scope of this Technical Report.

9.4.4 Protocol for application association establishment

9.4.4.1 Protocol for the establishment of confirmed application associations

AA establishment using the A-Associate service of the ACSE is the key element of DLMS/COSEM interoperability. The participants of an AA are:

- a client AP, proposing an AA; and
- a server AP, accepting the proposed AA or not.

NOTE 1 To support multicast and broadcast services, an AA can also be established between a client AP and a group of server APs.

Figure 39 gives the MSC for the case, when:

- the COSEM-OPEN.request primitive requests a confirmed AA;
- the connection of the supporting layers is required for the establishment of this AA.

A client AP that desires to establish a confirmed AA, invokes the COSEM-OPEN.request primitive of the ASO with Service_Class == Confirmed. Note, that the PH layer has to be connected before the COSEM-OPEN service is invoked. The response-allowed parameter of the xDLMS InitiateRequest APDU is set to TRUE. The client AL waits for an AARE APDU, prior to generating the .confirm primitive, with a positive – or negative – result.

The client CF enters the ASSOCIATION PENDING state. It examines then the Protocol_Connection_Parameters parameter. If this indicates that the establishment of the supporting layer connection is required, it establishes the connection. The CF assembles then –with the help of the xDLMS ASE and the ACSE – the AARQ APDU containing the parameters of the COSEM-OPEN.request primitive received from the AP and sends it to the server.

The CF of the server AL gives the AARQ APDU received to the ACSE. It extracts the ACSE related parameters then gives back the control to the CF. The CF passes then the contents of the user-information parameter of the AARQ APDU – carrying an xDLMS InitiateRequest APDU – to the xDLMS ASE. It retrieves the parameters of this APDU, then gives back the control to the CF. The CF generates the COSEM-OPEN.indication to the server AP with the parameters received APDU and enters the ‘ASSOCIATION PENDING’ state.

NOTE 2 Some service parameters of the COSEM-OPEN.indication primitive (address information, User_Information) do not come from the AARQ APDU, but from the supporting layer frame carrying the AARQ APDU. In some communication profiles, the Service_Class parameter of the COSEM-OPEN service is linked to the frame type of the supporting layer. In some other communication profiles, it is linked to the response-allowed field of the xDLMS Initiate.request APDU. See also 10.

NOTE 3 The ASEs only extract the parameters; their interpretation and the decision whether the proposed AA can be accepted or not is the job of the server AP.

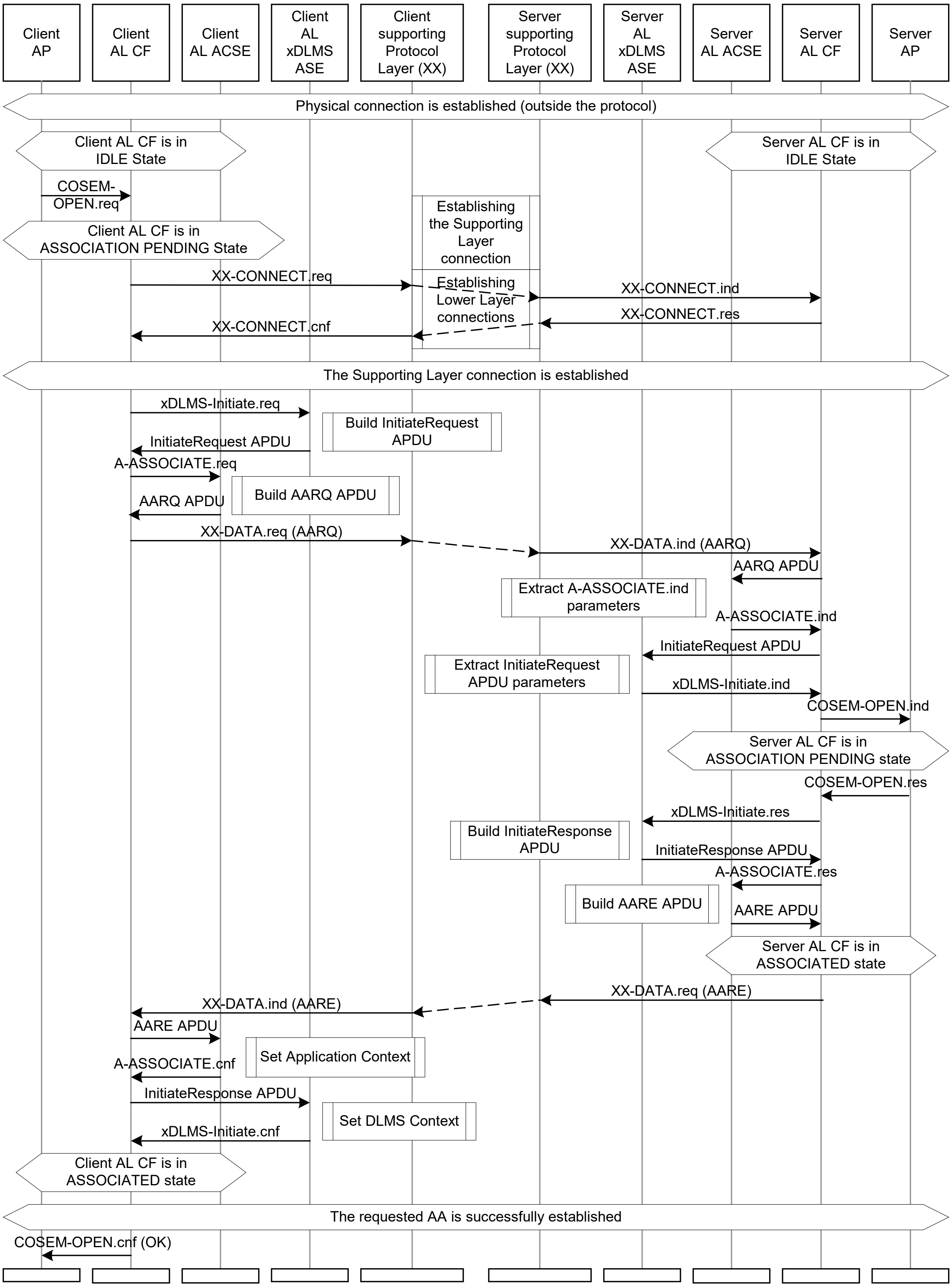

Client AP
Client AL CF
Client AL ACSE
Client AL xDLMS ASE
Client supporting Protocol Layer (XX)
Server supporting Protocol Layer (XX)
Server AL xDLMS ASE
Server AL ACSE
Server AL CF
Server AP
Physical connection is established (outside the protocol)
Client AL CF is in IDLE State
Server AL CF is in IDLE State
COSEM-OPEN.req
Establishing the Supporting Layer connection
Client AL CF is in ASSOCIATION PENDING State
XX-CONNECT.req
Establishing Lower Layer connections
XX-CONNECT.ind
XX-CONNECT.res
XX-CONNECT.cnf
The Supporting Layer connection is established
xDLMS-Initiate.req
Build InitiateRequest APDU
InitiateRequest APDU
A-ASSOCIATE.req
Build AARQ APDU
AARQ APDU
XX-DATA.req (AARQ)
XX-DATA.ind (AARQ)
AARQ APDU
Extract A-ASSOCIATE.ind parameters
A-ASSOCIATE.ind
InitiateRequest APDU
Extract InitiateRequest APDU parameters
xDLMS-Initiate.ind
COSEM-OPEN.ind
Server AL CF is in ASSOCIATION PENDING state
COSEM-OPEN.res
xDLMS-Initiate.res
Build InitiateResponse APDU
InitiateResponse APDU
A-ASSOCIATE.res
Build AARE APDU
AARE APDU
Server AL CF is in ASSOCIATED state
XX-DATA.req (AARE)
XX-DATA.ind (AARE)
AARE APDU
Set Application Context
A-ASSOCIATE.cnf
InitiateResponse APDU
Set DLMS Context
xDLMS-Initiate.cnf
Client AL CF is in ASSOCIATED state
The requested AA is successfully established
COSEM-OPEN.cnf (OK)

Figure 39 – MSC for successful AA establishment preceded by a successful lower layer connection establishment

The server AP parses the fields of the AARQ APDU as described below.

Fields of the Kernel functional unit:

- application-context-name: it carries the COSEM_Application_Context_Name the client proposes for the association;
- calling-AP-title: when the proposed application context uses ciphering, it shall carry the client system title. If a client system title has already been sent during a registration process, like in the S-FSK PLC profile, the calling-AP-title field should carry the same system title. Otherwise, the AA should be rejected and appropriate diagnostic information should be sent;
- calling_AE_invocation_identifier: this field supports the client user identification process; see DLMS UA 1000-1 Ed. 13:2019;
- calling-AE-qualifier: This field can be used to transport the public key certificate of the digital signature key of the client.

Fields of the authentication functional unit (when present):

- sender-acse-requirements:
  a) if not present or present but bit 0 = 0, then the authentication functional unit is not selected. Any following fields of the authentication functional unit may be ignored;

  b) if present and bit 0 = 1 then the authentication functional unit is selected;
- mechanism-name: it carries the COSEM_Authentication_Mechanism_Name the client proposes for the association;
- calling-authentication-value: it carries the authentication value generated by the client.

If the value of the mechanism-name or the calling-authentication-value fields are not acceptable then the proposed AA shall be refused.

When the parsing of the fields of the Kernel and the authentication functional unit is completed, the server continues with parsing the parameters of the xDLMS InitiateRequest APDU, carried by the user-information field of the AARQ:

- dedicated-key: it carries the dedicated key to be used in the AA being established;
- response-allowed: If the proposed AA is confirmed and the value of this parameter is TRUE (default), the server shall send back an AARE APDU. Otherwise, the server shall not respond. See also Clause 10;
- proposed-dlms-version-number, see 9.1.4.6;
- proposed-conformance, see 9.4.6.1;
- client-max-receive-pdu-size, see 9.1.4.8.

If all elements of the proposed AA are acceptable, the server AP invokes the COSEM-OPEN.response service primitive with the following parameters:

- Application_Context_Name: the same as the one proposed;
- Result: accepted;
- Failure_Type: Result-source: acse-service-user; Diagnostic: null;
- Responding_AP_Title: if the negotiated application context uses ciphering, it shall carry the server system title. If a server system title has already been sent during a registration process, like in the case of the S-FSK PLC profile, the Responding_AP_Title parameter should carry the same system title. Otherwise, the AA should be aborted by the client;
- Responding_AE_Qualifier: This field can be used to transport the public key certificate of the digital signature key of the server;
- Fields of the AARE authentication functional unit:
  - (Responder_)ACSE_Requirements:
    i) when no security (Lowest Level Security) authentication or Low Level Security (LLS) authentication is used, this field shall not be present, or if present, bit 0 (authentication) shall be set to 0. Any following fields of the authentication functional unit may be ignored;

    ii) when High Level Security (HLS) authentication is used, this field shall be present and bit 0 (authentication) shall be set to 1;
  - Security_Mechanism_Name: it shall carry the COSEM_Authentication_Mechanism_Name negotiated;
  - Responding_Authentication_Value: it carries the authentication value generated by the server (StoC).
- Negotiated_xDLMS_Context.

The CF assembles the AARE APDU – with the help of the xDLMS ASE and the ACSE – and sends it to the client AL via the supporting layer protocols, and enters the ASSOCIATED state. The proposed AA is established now; the server is able to receive xDLMS data transfer service request(s) – both confirmed and unconfirmed – and to send responses to confirmed service requests within this AA.

At the client side, the fields of the AARE APDU received are extracted with the help of the ACSE and the xDLMS ASE, and passed to the client AP via the COSEM-OPEN.confirm service primitive. At the same time, the client AL enters the 'ASSOCIATED' state. The AA is established now with the application context and xDLMS context negotiated.

If the application context proposed by the client is not acceptable or the authentication of the client is not successful, the COSEM-OPEN.response primitive is invoked with the following parameters:

- Application_Context_Name: the same as the one proposed, or the one supported by the server;
- Result: rejected-permanent or rejected-transient;
- Failure_Type: Result-source: acse-service-user; Diagnostic: an appropriate value;
- User_Information: an xDLMS InitiateResponse APDU with the parameters of the xDLMS context supported by the server.

If the application context proposed by the client is acceptable and the authentication of the client is successful but the xDLMS context cannot be accepted, the COSEM-OPEN.response primitive shall be invoked with the following parameters:

- Application_Context_Name: the same as the one proposed;
- Result: rejected-permanent or rejected-transient;
- Failure_Type: Result-source: acse-service-user; Diagnostic: no-reason-given;
- xDLMS_Initiate_Error, indicating the reason for not accepting the proposed xDLMS context.

In these two cases, upon the invocation of the .response primitive, the CF assembles and sends the AARE APDU to the client via the supporting layer protocols. The proposed AA is not established, the server CF returns to the IDLE state.

At the client side, the fields of the AARE APDU received are extracted with the help of the ACSE and the xDLMS ASE, and passed to the client AP via the COSEM-OPEN.confirm primitive. The proposed AA is not established, the client CF returns to the IDLE state.

The server ACSE may not be capable of supporting the requested association, for example if the AARQ syntax or the ACSE protocol-version are not acceptable. In this case, it returns a COSEM-OPEN.response primitive to the client with an appropriate Result parameter. The result-source-diagnostic field of the AARE APDU is appropriately assigned the symbolic value of “acse- service-provider”. The COSEM-OPEN.indication primitive is not issued. The association is not established.

For more information see the complete Green Book.

9.4.5 Protocol for application association release

9.4.5.1 Overview

An existing AA can be released gracefully or non-gracefully. Graceful release is initiated by the client AP. Non-graceful release takes place when an event unexpected by the AP occurs, for example a physical disconnection is detected.

9.4.5.2 Graceful release of an application association

DLMS/COSEM provides two mechanisms to release AAs:

- by disconnecting the supporting layer of the AL;
- by using the ACSE A-Release service.

The first mechanism shall be supported in all profiles where the supporting layer of the AL is connection oriented.

For more information see the complete Green Book.

9.4.6 Protocol for the data transfer services

9.4.6.1 Negotiation of services and options – the conformance block

The conformance block allows clients and servers using the same DLMS/COSEM protocol, but supporting different capabilities to negotiate a compatible set of capabilities so that they can communicate. It is carried by the DLMS_Conformance parameter of the COSEM-OPEN service.

In DLMS/COSEM none of the services or options are mandatory: the ones to be used are negotiated via the COSEM-OPEN service (the proposed-conformance parameter of the xDLMS InitiateRequest APDU and the negotiated-conformance parameter of the xDLMS InitiateResponse APDU). An implemented service shall be fully conforming to its specification. If a service or option is not present in the negotiated conformance block, it should not be requested by the client.

For more information see the complete Green Book.

9.4.6.2 Confirmed and unconfirmed xDLMS service invocations

In general, xDLMS services may be invoked in a confirmed or an unconfirmed manner. The time sequence of the service primitives corresponds to:

- Figure 35 item a) in the case of confirmed service invocations; and
- Figure 35 item d) in the case of unconfirmed service invocations.

A client AP that desires to access an attribute or a method of a COSEM object invokes the appropriate .request service primitive. The client AL constructs the APDU corresponding to the .request primitive and sends it to the server.

The server AP, upon the receipt of the .indication primitive, checks whether the service can be provided or not (validity, client access rights, availability, etc.). If everything is OK, it locally applies the service required on the corresponding “real” object. In the case of confirmed services, the server AP invokes the appropriate .response primitive. The server AL constructs the APDU corresponding to the .response primitive and sends it to the server. The client AL generates the .confirm primitive.

If a confirmed service request cannot be processed by the server AL – for example the request has been received without establishing an AA first, or the request is otherwise erroneous – it is either discarded, or when possible, the server AL responds with a confirmedServiceError APDU, or, when implemented, with an ExceptionResponse APDU. These APDUs may contain diagnostic information about the reason of not being able to process the request. They are defined in 9.5.

Within confirmed AAs xDLMS services can be invoked in a confirmed or unconfirmed manner.

Within unconfirmed AAs xDLMS services may be invoked in an unconfirmed manner only. With this, collisions due to potential multiple responses in the case of multicasting and/or broadcasting can be avoided.

For more information see the complete Green Book.

9.4.6.3 Protocol for the GET service

When the client AP desires to read the value of one or more COSEM object attributes, it uses the GET service.

As explained in 9.3.6, the encoded form of the request shall always fit in a single APDU.

On the other hand, the result may be too long to fit in a single APDU. In this case, either the service-specific or the general block transfer mechanism may be used. It is negotiated via bit 2 or bit 11 of the conformance block, see 9.4.6.1.

NOTE In some DLMS/COSEM communication profiles segmentation is available to transfer long APDUs.

For more information see the complete Green Book.

9.4.6.4 Protocol for the SET service

When the client AP desires to write the value of one or more COSEM object attributes, it uses the SET service.

As explained in 9.3.7, the encoded form of the request may fit in a single request or not. In this latter case, either the service-specific or the general block transfer mechanism may be used. It is negotiated via bit 2 or bit 12 of the conformance block, see 9.4.6.1.

NOTE 1 In some DLMS/COSEM communication profiles segmentation is available to transfer long APDUs.

For more information see the complete Green Book.

9.4.6.5 Protocol for the ACTION service

When the client AP desires to invoke one or more COSEM objects methods, it uses the ACTION service. As explained in 9.3.8, the ACTION service comprises two phases.

If the method references and method invocation parameters or the return parameters do not fit in a single APDU, either the service-specific or the general block transfer mechanism may be used. It is negotiated via bit 2 or bit 13 of the conformance block, see 9.4.6.1.

NOTE In some DLMS/COSEM communication profiles segmentation is available to transfer long APDUs.

For more information see the complete Green Book.

9.4.6.6 Protocol for the ACCESS service

The client can use the ACCESS service to read or write the value of one or more COSEM object attributes or to invoke one or more methods.

The protocol of the ACCESS service is specified by way of message sequence charts, including cases where it is used together with general block transfer and general message protection.

For more information see the complete Green Book.

9.4.6.7 Protocol of the DataNotification service

When the server AP invokes a DataNotification.request service primitive, the server AL builds the DataNotification APDU and sends it to the client.

When the client AL receives this APDU, it invokes the DataNotification.indication service primitive.

If the service primitives are long partial service invocations can be used.

If the encoded form of the service primitive is too long, then the general block transfer mechanism can be used.

For more information see the complete Green Book.

9.4.6.8 Protocol for the EventNotification service

Upon invocation of the EventNotification.request service, the Server AL builds an event-notification-request APDU. The possibilities to send out this APDU depend on the communication profile and the connection status of the lower layers. Therefore, the protocol of the EventNotification service is further discussed in 10.

For more information see the complete Green Book.

9.4.6.9 Protocol for the Read service

As explained in 9.3.14, the Read service is used when the server uses SN referencing, either to read (a) COSEM object attribute(s), or to invoke (a) method(s) when return parameters are expected:

- in the first case, the GET.request service primitives are mapped to Read.request primitives and the Read.confirm primitives are mapped to GET.confirm primitives.
- in the second case, the ACTION.request service primitives are mapped to Read.request primitives and the Read.response primitives are mapped to ACTION.response primitives.
- *For more information see the complete Green Book.*

9.4.6.10 Protocol for the Write service

As explained in 9.3.15, the Write service is used when the server uses SN referencing, either to write (a) COSEM object attribute(s), or to invoke (a) method(s) when no return parameters are expected:

- in the first case, the SET.request service primitives are mapped to Write.request primitives and the Write.confirm primitives to SET.confirm primitives.
- in the second case, the ACTION.request service primitives are mapped to Write.request primitives and the Write.response primitives to ACTION.confirm primitives.

For more information see the complete Green Book.

9.4.6.11 Protocol for the UnconfirmedWrite service

This service may be invoked only when an AA has already been established. Depending on the communication profile, the APDU corresponding to the request may be transported using the connection-oriented or connectionless data services of the supporting protocol layer.

As explained in 9.3.16, the UnconfirmedWrite service may be used either to write (a) COSEM object attribute(s), or to invoke (a) method(s) when no return parameters are expected:

- in the first case, the SET.request service primitives are mapped to UnconfirmedWrite.request primitives.
- in the second case, the ACTION.request service primitives are mapped to UnconfirmedWrite.request primitives.

For more information see the complete Green Book.

9.4.6.12 Protocol for the InformationReport service

The protocol for the InformationReport service, specified 9.3.17 in is essentially the same as that of the EventNotification service, 9.4.6.6.

As, unlike the EventNotification service, the InformationReport service does not contain the optional Application_Addresses parameter, the information report is always sent by the Server Management Logical Device to the Client Management AP.

For more information see the complete Green Book.

9.4.6.13 Protocol of general block transfer mechanism

9.4.6.13.1 Introduction

The general block transfer (GBT) mechanism can be used to carry any xDLMS APDU when the service parameters are long i.e. their encoded form with the protection overhead, if any, is longer than the Max Receive PDU Size of the peer negotiated. In this case, the AL uses one or more General-Block-Transfer (GBT) xDLMS APDUs to transport such long APDUs.

The service primitive invocations may be complete including all the service parameters, or partial including only one part of the service parameters. Using complete or partial service invocations is left to the implementation.

Following the reception of a service .request / .response service primitive from the AP, the AL:

- builds the APDU that carries the service primitive;
- when ciphering is required it applies the protection as required by the Security_Options and builds the appropriate ciphered APDU;
- when the resulting APDU is longer than the negotiated max APDU size, then the AL uses the GBT mechanism to send the complete message in several GBT APDUs.

However, there is no direct relationship between partial invocations and the GBT APDUs sent. The AL may apply the protection using complete or partial service invocations.

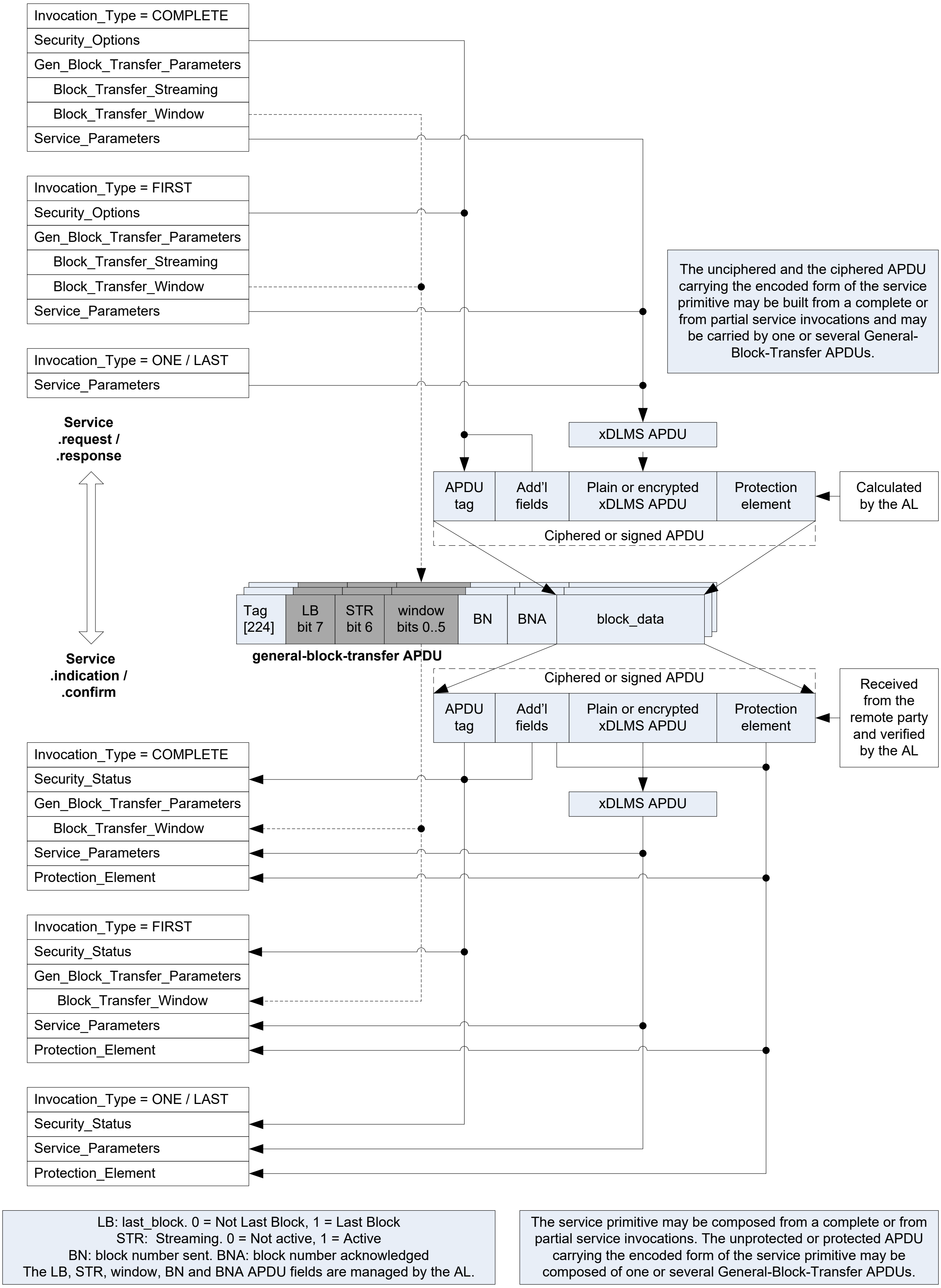
Following the reception of GBT APDUs from a remote party, the AL:

- assembles the block-data fields of the GBT APDUs received together;
- when the resulting complete APDU is ciphered, it checks and removes the protection;
- it invokes the appropriate service primitive, passing the additional Security_Status, the General_Block_Transfer_Parameters and the Protection_Element.

However, there is no direct relationship between the GBT APDUs received and the partial service invocations. The AL may verify and remove the protection processing the GBT APDUs or processing the complete, assembled APDU.

See also Figure 36.

A confirmed message exchange may be started without or with using GBT. However, if one party sends a request or a response using GBT, the other party shall follow. The parties continue then using GBT until the end, i.e. until the complete response will have been received.


Invocation_Type = COMPLETE
Security_Options
Gen_Block_Transfer_Parameters
Block_Transfer_Streaming
Block_Transfer_Window
Service_Parameters
Invocation_Type = FIRST
Security_Options
Gen_Block_Transfer_Parameters
Block_Transfer_Streaming
Block_Transfer_Window
Service_Parameters
Invocation_Type = ONE / LAST
Service_Parameters
The unciphered and the ciphered APDU carrying the encoded form of the service primitive may be built from a complete or from partial service invocations and may be carried by one or several General-Block-Transfer APDUs.
Service .request / .response
xDLMS APDU
APDU tag
Add'l fields
Plain or encrypted xDLMS APDU
Protection element
Calculated by the AL
Ciphered or signed APDU
Tag [224]
LB bit 7
STR bit 6
window bits 0..5
BN
BNA
block_data
general-block-transfer APDU
Service .indication / .confirm
Ciphered or signed APDU
APDU tag
Add'l fields
Plain or encrypted xDLMS APDU
Protection element
Received from the remote party and verified by the AL
xDLMS APDU
Invocation_Type = COMPLETE
Security_Status
Gen_Block_Transfer_Parameters
Block_Transfer_Window
Service_Parameters
Protection_Element
Invocation_Type = FIRST
Security_Status
Gen_Block_Transfer_Parameters
Block_Transfer_Window
Service_Parameters
Protection_Element
Invocation_Type = ONE / LAST
Security_Status
Service_Parameters
Protection_Element
LB: last_block. 0 = Not Last Block, 1 = Last Block
STR: Streaming. 0 = Not active, 1 = Active
BN: block number sent. BNA: block number acknowledged
The LB, STR, window, BN and BNA APDU fields are managed by the AL.
The service primitive may be composed from a complete or from partial service invocations. The unprotected or protected APDU carrying the encoded form of the service primitive may be composed of one or several General-Block-Transfer APDUs.

NOTE Applying and checking/removing cryptographic protection on APDUs is independent from the GBT process. It is included here for completeness.

Figure 40 – Partial service invocations and GBT APDUs

Streaming of blocks is managed by the AL taking into account the GBT parameters passed from the local AP to the AL – see 9.3.5 – and the fields of GBT APDUs – see Figure 40 – received from the remote AL.

The various service invocation types – COMPLETE, FIRST-PART, ONE-PART or LAST-PART – and the relationship between these invocations, the service parameters and the fields of the ciphered APDUs and the General-Block-Transfer (GBT) APDUs are shown in Figure 40.

The Block_Transfer_Streaming (BTS) parameter is passed by the AP to the AL to indicate that the AL can send blocks in streams, i.e. without waiting for a confirmation of each block received by the remote party. This parameter is not included in the APDU.

The Block_Transfer_Window (BTW) parameter indicates the size of the streaming window supported, i.e. the maximum number of blocks that can be received. The Block_Transfer_Window parameter of the other party may be known *a priori* by the parties. However, the window size is managed by the AL: it can use a lower value, for example during lost block recovery.

NOTE 1 This relationship is indicated using a dotted line in Figure 40 between the Block_Transfer_Window parameter and the window field of the APDU.

In the case of unconfirmed services the Block_Transfer_Streaming parameter shall be set to FALSE and the Block_Transfer_Window shall be set to 0. This indicates to the AL that it shall send the encoded form of the whole service primitive in as many GBT APDUs as needed without waiting for confirmation of the blocks sent.

For more information see the complete Green Book.

9.4.6.13.2 The GBT procedure

9.4.6.13.2.1 Overview

The GBT procedure is shown in Figure 41. It can be:

- confirmed, to carry APDUs corresponding to the service primitives of any confirmed xDLMS service;
- unconfirmed, to carry an APDU corresponding to an unconfirmed service primitive from the client to the server or to carry an APDU corresponding to an unsolicited service request from the server to the client.

  NOTE GBT is often used with the DataNotification and Access services.

The GBT procedure is essentially the same on the client and on the server side. It is called by the AL when:

a) a service primitive – .request or .response – is invoked by the AP and either:

   - Invocation_Type = COMPLETE or FIRST-PART with the GBT parameters BTS and BTW present; or
   - the encoded and cryptographically protected APDU to be sent is too long to fit into the maximum APDU size negotiated;

b) a GBT APDU is received from the peer.

The GBT procedure comprises three sub-procedures:

1) *Send GBT APDU stream,* see the complete Green Book;
2) *Process GBT APDU,* see 9.4.6.13.5;
3) *Check RQ and fill gaps,* see the complete Green Book.

These sub-procedures are called by the AL as described in the complete Green Book.

The model of the GBT procedure uses a Send Queue SQ and a Receive Queue RQ. Each block B in the queues has a block number BN, a flag LastBlock LB and a payload BlockData BD (that may be empty). The queues are ordered by BN.

The SQ is filled by the AL after processing the service primitives invoked by the AP i.e.:

- building the APDU;
- applying cryptographic protection as stipulated by the protection parameters;
- splitting the protected APDU into blocks.

For all service primitives, partial service invocations may be used as specified in 9.3.5, Figure 36 and 9.4.6.13, Figure 40. However, this has no consequence for the GBT protocol. The service Invocation_Type may be COMPLETE, FIRST-PART, ONE-PART or LAST-PART. Each service invocation adds one or more blocks to the SQ. The block number BN is incremented every time a new block is added.

The blocks in the SQ are sent by the *Send GBT APDU stream* sub-procedure.

The RQ is filled by the *Process GBT APDU* sub-procedure. Gaps in the RQ – in the case of a confirmed GBT procedure – are filled by the *Check RQ and fill gaps* sub-procedure.

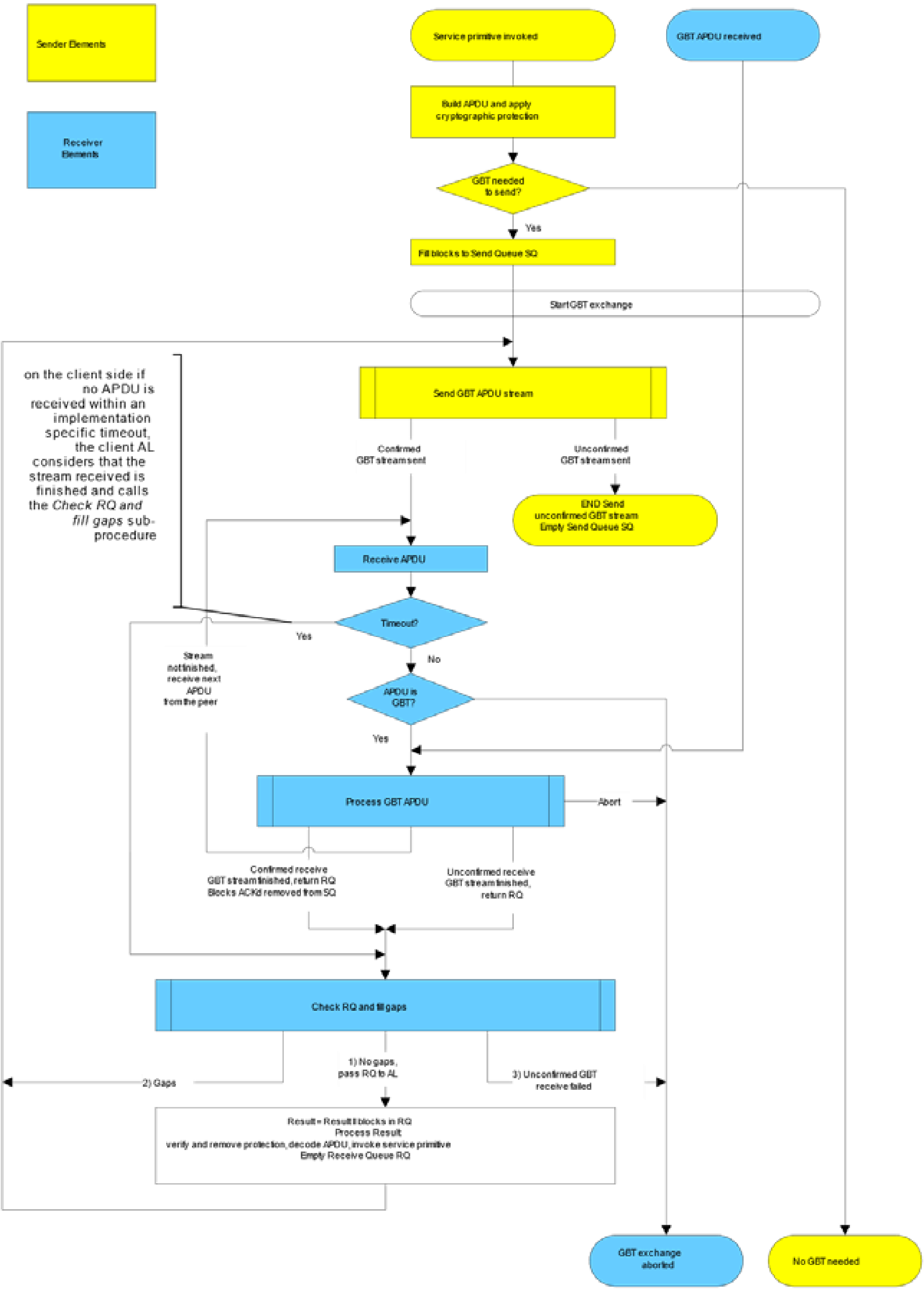

Sender Elements
Receiver Elements
Service primitive invoked
GBT APDU received
Build APDU and apply cryptographic protection
GBT needed to send?
Yes
Fill blocks to Send Queue SQ
Start GBT exchange
on the client side if no APDU is received within an implementation specific timeout, the client AL considers that the stream received is finished and calls the *Check RQ and fill gaps* sub-procedure
Send GBT APDU stream
Confirmed GBT stream sent
Unconfirmed GBT stream sent
END Send unconfirmed GBT stream Empty Send Queue SQ
Receive APDU
Timeout?
Yes
No
Stream not finished, receive next APDU from the peer
APDU is GBT?
Yes
Process GBT APDU
Abort
Confirmed receive GBT stream finished, return RQ Blocks ACK'd removed from SQ
Unconfirmed receive GBT stream finished, return RQ
Check RQ and fill gaps
2) Gaps
1) No gaps, pass RQ to AL
3) Unconfirmed GBT receive failed
Result = Result||blocks in RQ
Process Result
verify and remove protection, decode APDU, invoke service primitive
Empty Receive Queue RQ
GBT exchange aborted
No GBT needed

Figure 41 – The GBT procedure

For more information see the complete Green Book.

9.4.6.13.5 Process GBT APDU sub-procedure

9.4.6.13.5.1 General

The *Process GBT APDU* sub-procedure is shown in Figure 42.

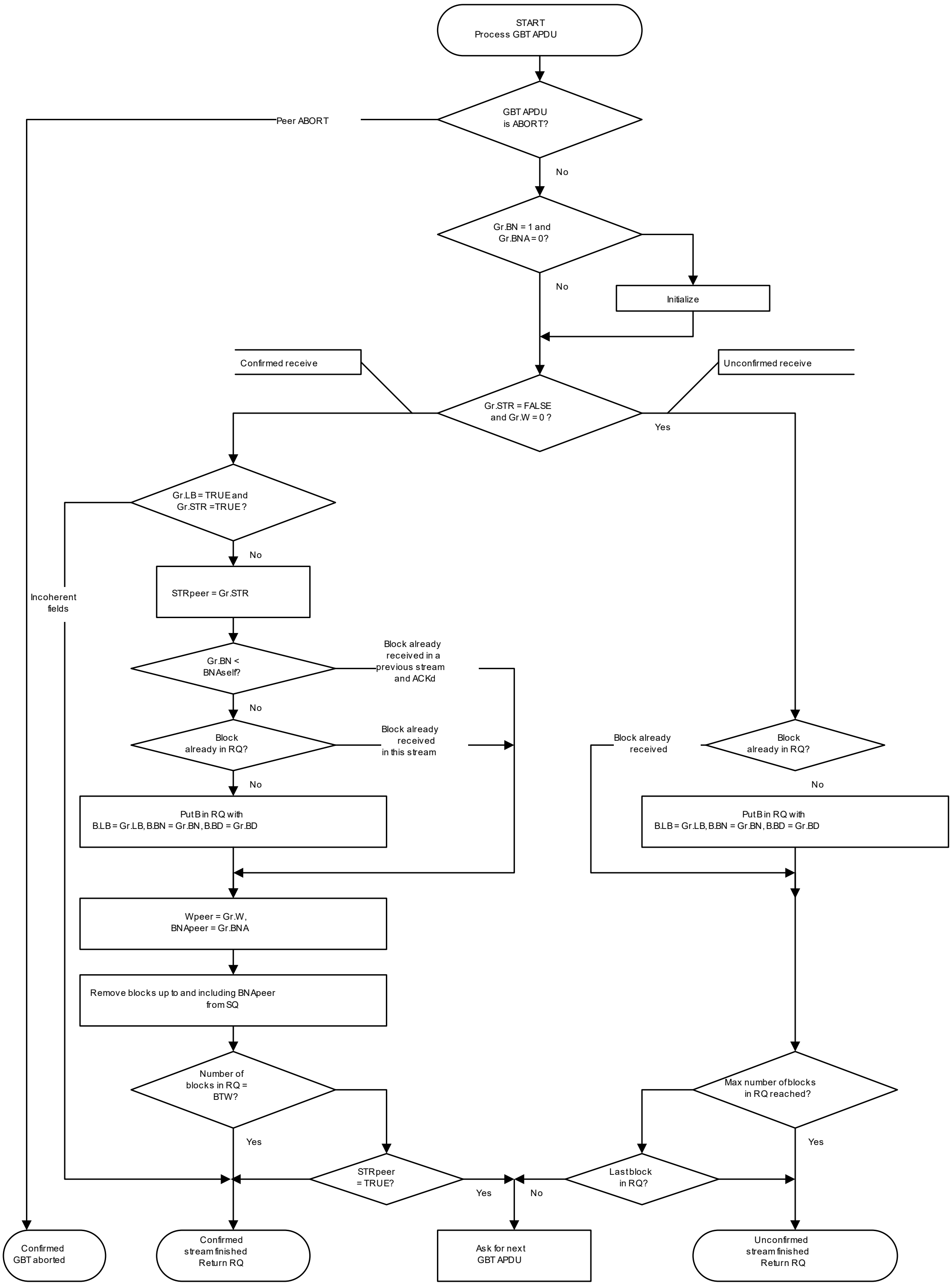


Figure 42 – Process GBT APDU sub-procedure

First, the value of Gr.STR and Gr.BN is checked. If the GBT APDU is an ABORT GBT APDU – see the complete Green Book– this indicates that the peer wants to abort the exchange.

Then, the value of Gr.BN and Gr.BNA is checked. If Gr.BN = 1 and Gr.BNA = 0, this indicates the start of a new GBT exchange and the initial values shall be set according to the complete Green Book.

Then, the value of Gr.STR and Gr.W is checked. If Gr.STR = FALSE and Gr.W = 0, this indicates that the GBT procedure is unconfirmed; otherwise it is confirmed.

For more information see the complete Green Book.

9.5 Abstract syntax of COSEM PDUs

The abstract syntax of COSEM PDUs is specified in this subclause using ASN.1. See ISO/IEC 8824:2008.

NOTE The CIASE APDUs are specified in the complete Green Book.

```
COSEMpdu DEFINITIONS ::= BEGIN

ACSE-APDU ::= CHOICE
{
    aarq                              AARQ-apdu,
    aare                              AARE-apdu,
    rlrq                              RLRQ-apdu,          -- OPTIONAL
    rlre                              RLRE-apdu           -- OPTIONAL
}
```

For more information see the complete Green Book.

9.6 COSEM PDU XML schema

9.6.1 General

ITU-T recommendations X.693 and X.694 provide XML encoding rules to Abstract Syntax Notation 1 (ASN.1) and XML Schema Definitions Language (XSD) mapping to ASN.1. No recommendation is provided to map ASN.1 to XSD.

XML has gained wide acceptance in the IT industry. The purpose of such encoding is to enable transfer of COSEM model content with various means in the form of XML encoded content. It can be a XML document exchanged between applications, content included in Web services SOAP messages or content encapsulated in e-mail messages, to name just few of the applications.

Interoperability and mapping between ASN.1 encoded APDUs and XML encoded content is important in both directions. On one side ASN.1 has enabled creation of XML encoded content with support for XML Encoding Rules (See ITU-T X.693 and ITU-T X.694). On the other hand, IT industry is searching for solutions for optimal transfer of XML content with XML optimized packaging. For that purposes conversion between W3C XML Schema and ASN.1 definition is crucial.

Conversion in both directions enables proper conversion of XML content to ASN.1 encoded content and vice versa. Subclause 9.6.2 contains mapping of COSEMPdu ASN.1 definition into COSEMPdu XML Schema (XSD).

9.6.2 XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns="http://www.dlms.com/COSEMpdu"
      targetNamespace="http://www.dlms.com/COSEMpdu"
      elementFormDefault="qualified">

  <!-- ASN.1 definitions -->
  <xsd:complexType name="NULL" final="#all" />

  <xsd:simpleType name="BitString">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-1]{0,}" />
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="ObjectIdentifier">
    <xsd:restriction base="xsd:token">
      <xsd:pattern value="[0-2](\.[1-3]?[0-9]?(\.\d+)*)?" />
    </xsd:restriction>
  </xsd:simpleType>
```

For more information see the complete Green Book.

10 Using the DLMS/COSEM application layer in various communications profiles

10.1 Communication profile specific elements

10.1.1 General

The COSEM interface model for energy metering equipment, specified in DLMS UA 1000-1 has been designed for use with a variety of communication profiles for exchanging data over various communication media. In each such profile, the application layer is the DLMS/COSEM AL, providing the xDLMS services to access attributes and methods of COSEM objects. For each communication profile, the following elements must be specified:

- the targeted communication environments;
- the structure of the profile (the set of protocol layers);
- the identification/addressing scheme;
- mapping of the DLMS/COSEM AL services to the service set provided and used by the supporting layer;
- communication profile specific parameters of the DLMS/COSEM AL services;
- other specific considerations/constraints for using certain services within a given profile.

10.1.2 Targeted communication environments

This part identifies the communication environments, for which the given communication profile is specified.

10.1.3 The structure of the profile

This part specifies the protocol layers included in the given profile.

10.1.4 Identification and addressing schemes

This part describes the identification and addressing schemes specific for the profile.

As described in DLMS UA 1000-1 Ed. 13:2019, 4.1.7, metering equipment is modelled in COSEM as physical devices, containing one or more logical devices. In the COSEM client/server type model, data exchange takes place within AAs, between a COSEM client AP and a COSEM Logical Device, playing the role of a server AP.

To be able to establish the required AA and then exchanging data with the help of the supporting layer protocols, the client- and server APs must be identified and addressed, according to the rules of a communication profile. At least the following elements need to be identified / addressed:

- physical devices hosting clients and servers;
- client- and server APs;

The client- and server APs also identify the AAs.

10.1.5 Supporting layer services and service mapping

This part specifies the service mapping between the services requested by the DLMS/COSEM AL and the services provided by its supporting layer.

In each communication profile, the DLMS/COSEM AL provides the same set of services to the client- and server APs. However, the supporting protocol layer in the various profiles provides a different set of services to the service user AL.

The service mapping specifies how the AL is using the services of its supporting layer to provide ACSE and xDLMS services to its service user. For this purpose generally MSCs are used showing the sequence of the events following a service invocation by the COSEM AP.

10.1.6 Communication profile specific parameters of the DLMS/COSEM AL services

In DLMS/COSEM, only the COSEM-OPEN service has communication profile specific parameters. Their values and use are defined as part of the communication profile specification.

10.1.7 Specific considerations / constraints using certain services within a given profile

The availability and the protocol of some of the services may depend on the communication profile. These elements are specified as part of the communication profile specification.

10.2 The 3-layer, connection-oriented, HDLC based communication profile

10.2.1 Targeted communication environments

The 3-layer, CO, HDLC based profile is suitable for local data exchange with metering equipment via direct connection, or remote data exchange via the PSTN or GSM networks.

10.2.2 The structure of the profile

This profile is based on a three-layer (collapsed) OSI protocol architecture:

- the DLMS/COSEM AL, specified in clause 9;
- the data link layer based on the HDLC standard, specified in Clause 8;
- the physical layer; specified in Clause 5. The use of the PhL for the purposes of direct local data exchange using an optical port or a current loop physical interface is specified in Clause 6.

10.2.3 Identification and addressing scheme

The HDLC based data link layer provides services to the DLMS/COSEM AL at Data Link SAP-s, also called as the Data Link- or HDLC addresses.

On the client side, only the client AP needs to be identified. The addressing of the physical device hosting the client APs is done by the PhL (for example by using phone numbers).

On the server side, several physical devices may share a common physical line (multidrop configuration). In the case of direct connection this may be a current loop as specified in IEC 62056-21. In the case of remote connection several physical devices may share a single telephone line. Therefore both the physical devices and the logical devices hosted by the physical devices need to be identified. This is done using the HDLC addressing mechanism as described in 8.4.2:

- physical devices are identified by their lower HDLC address;
- logical devices within a physical device are identified by their upper HDLC address;
- a COSEM AA is identified by a doublet, containing the identifiers of the two APs participating in the AA.

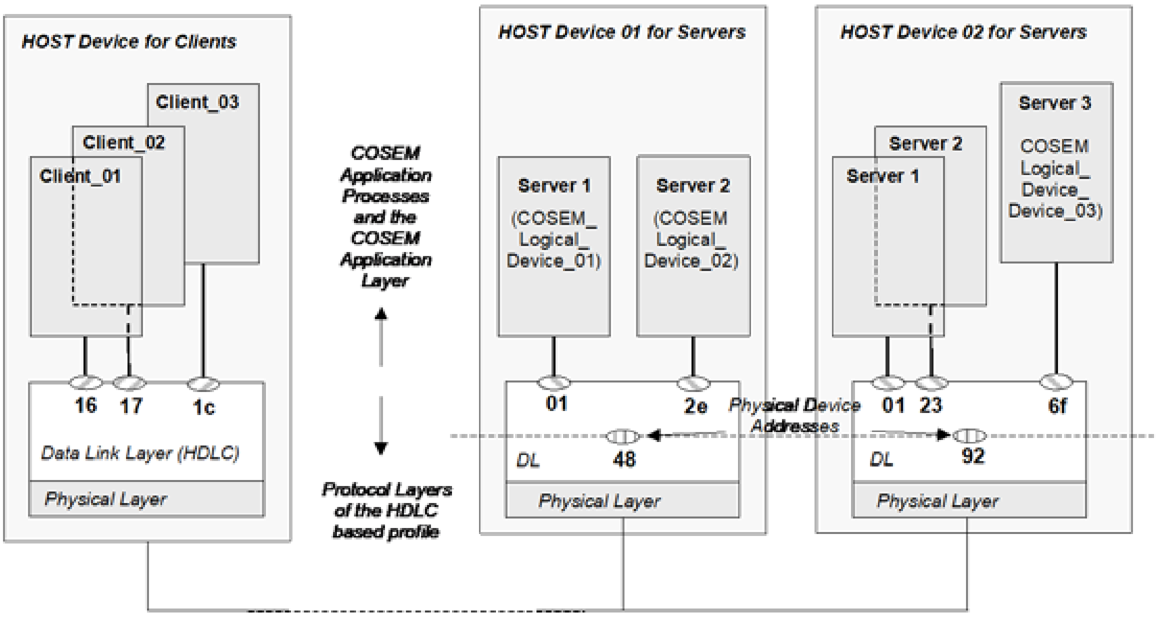

HOST Device for Clients
Client_03
Client_02
Client_01
16
17
1c
Data Link Layer (HDLC)
Physical Layer
COSEM Application Processes and the COSEM Application Layer
Protocol Layers of the HDLC based profile
HOST Device 01 for Servers
Server 1 (COSEM_ Logical_ Device_01)
Server 2 (COSEM Logical_ Device_02)
01
2e
DL
48
Physical Layer
Physical Device Addresses
HOST Device 02 for Servers
Server 3
COSEM Logical_ Device_ Device_03)
Server 2
Server 1
01
23
6f
DL
92
Physical Layer

Figure 43 – Identification/addressing scheme in the 3-layer, CO, HDLC based communication profile

For example, an AA between Client_01 (HDLC address = 16) and Server 2 in Host Device 02 (HDLC address = 2392) is identified by the doublet {16, 2392}. Here, “23” is the upper HDLC address and “92” is the lower HDLC address. All values are hexadecimal. This scheme ensures that a particular COSEM AP (client or server) may support more than one AA simultaneously without ambiguity. See Figure 43.

10.2.4 Supporting layer services and service mapping

In this profile, the supporting layer of the DLMS/COSEM AL is the HDLC based data link layer. It provides services for:

- data link layer connection management;
- connection-oriented data transfer;
- connection-less data transfer.

Figure 44 summarizes the data link layer services provided for and used by the DLMS/COSEM AL.

The DL-DATA.confirm primitive on the server side is available to support transporting long messages from the server to the client in a transparent manner to the AL. See 10.2.6.5.

In some cases, the correspondence between an AL (ASO) service invocation and the supporting data link layer service invocation is straightforward. For example, invocation of a GET.request primitive directly implies the invocation of a DL-DATA.request primitive.

In some other cases a direct service mapping cannot be established. For example, the invocation of a COSEM-OPEN.request primitive with Service_Class == Confirmed involves a series of actions, starting with the establishment of the lower layer connection with the help of the DL-CONNECT service, and then sending out the AARQ APDU via this newly established connection using a DL-DATA.request service. Examples for service mapping are given in 9.4.

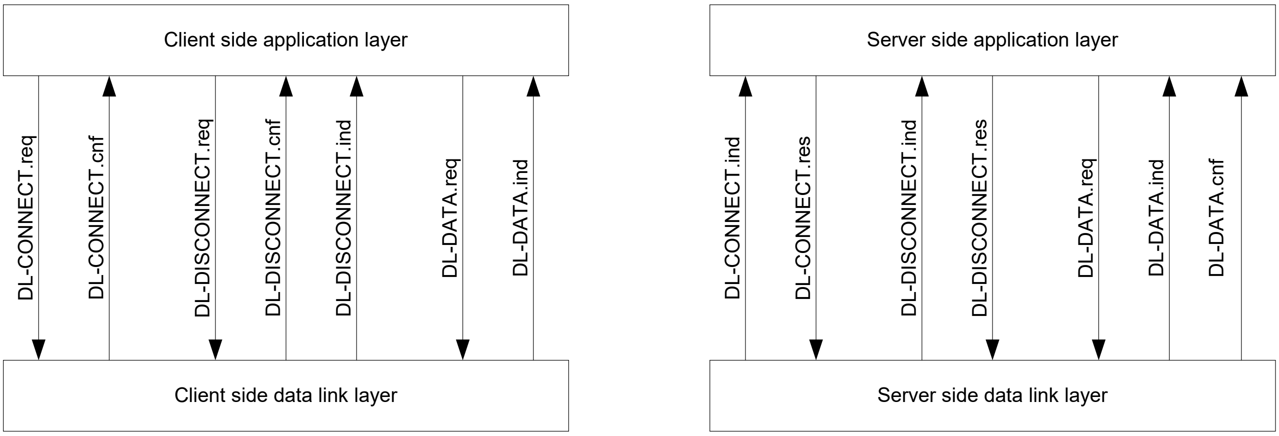


Figure 44 – Summary of data link layer services

10.2.5 Communication profile specific service parameters of the DLMS/COSEM AL services

Only the COSEM-OPEN service has communication profile specific parameters, the Protocol_Connection_Parameters parameter. This contains the following data:

- Protocol (Profile) Identifier 3-Layer, connection-oriented, HDLC based;
- Server_Lower_MAC_Address (COSEM Physical Device Address);
- Server_Upper_MAC_Address (COSEM Logical Device Address);
- Client_MAC_Address;
- Server_LLC_Address;
- Client_LLC_Address.

Any server (destination) address parameter may contain special addresses (All-station, No-station, etc.). For more information, see Clause 8.

10.2.6 Specific considerations / constraints

10.2.6.1 Confirmed and unconfirmed AAs and data transfer service invocations, frame types used

Table 15 summarizes the rules for establishing confirmed and unconfirmed AAs, the type of data transfer services available in such AAs and the HDLC frame types that carry the APDU-s. This table clearly shows one of the specific features of this profile: the Service_Class parameter of service invocations is linked to the frame type of the supporting layer:

- if the COSEM-OPEN service – see 9.3.2 – is invoked with Service_Class == Confirmed, then the AARQ APDU is carried by an “I” frame. On the other hand, if it is invoked with Service_Class == Unconfirmed it is carried by a “UI” frame. Therefore, in this profile, the response-allowed parameter of the xDLMS InitiateRequest APDU has no significance. See also 9.4.4.1;
- similarly, if a data transfer service .request primitive is invoked with Service_Class == Confirmed, then the corresponding APDU is transported by an “I” frame. If it is invoked with Service_Class == Unconfirmed then the corresponding APDU is carried by a “UI” frame. Consequently, service-class bit of the Invoke-Id-And-Priority / Long-Invoke-Id-And-Priority field – see 9.5 – is not relevant in this profile.

When the server is accessed via a gateway – see the complete Green Book – and the APDU is encrypted, the gateway is not able to check the response-allowed field of the xDLMS InitiateRequest APDU or the service-class bit of the Invoke-Id-And-Priority / Long-Invoke-Id-And-Priority field to determine if the APDU carries a confirmed or an unconfirmed service request.

Therefore, when between the gateway and the server the 3-layer, C.O. HDLC based profile is used, the gateway always places the APDU received to an I frame and forwards it to the server.

When the server receives an AARQ APDU carried by an I frame it shall check the response-allowed field of the xDLMS InitiateRequest APDU. If it is set to FALSE, it shall not respond.

In the case when the server receives an xDLMS APDU in an I frame it shall check the service-class bit of the Invoke-Id-And-Priority / Long-Invoke-Id-And-Priority field when this field is present. If it is set to 0, it shall not respond.

When the server receives an AARQ or an xDLMS APDU in a UI frame, it shall not respond.

Table 15 – Application associations and data exchange in the 3-layer, CO, HDLC based profile

Application association establishment				Data exchange	
Protocol connection parameters	COSEM-OPEN service class	Use	Type of established AA	Service class	Use
Id: HDLC LLC and MAC addresses	Confirmed	1/ Connect data link layer 2/ Exchange AARQ/AARE APDU-s transported in "I" frames	Confirmed	Confirmed	"I" frame
				Unconfirmed	"UI" frame
	Unconfirmed	Send AARQ in a "UI" frame	Unconfirmed	Confirmed (not allowed)	-
				Unconfirmed	"UI" frame
NOTE As described above this table, when the server is accessed via a gateway, the COSEM APDUs are always carried by I frames. The server has to check the response-allowed field of the xDLMS InitiateRequest APDU or the service-class bit of Invoke-Id-And-Priority / Long-Invoke-Id-And-Priority field as applicable to determine if the service request is confirmed or unconfirmed.					

10.2.6.2 Correspondence between AAs and data link layer connections, releasing AAs

In this profile, a confirmed AA is bound to a supporting data link layer connection, in a one-to-one basis. Consequently:

- establishing a confirmed AA implies the establishment of a connection between the client and server data link layers;
- a confirmed AA in this profile can be non-ambiguously released by disconnecting the corresponding data link layer connection.

On the other hand, in this profile establishing an unconfirmed AA does not need any lower layer connection: consequently, once established, unconfirmed AAs with servers not supporting the ACSE A-RELEASE service (see 9.3.3 and 9.4.5) cannot be released.

10.2.6.3 Service parameters of the COSEM-OPEN / -RELEASE / -ABORT services

Thanks to the possibility to transparently transport higher layer related information within the SNRM and DISC HDLC frames, this profile allows the use of the optional "User_Information" parameter of the COSEM-OPEN – see 9.3.2 – and COSEM-RELEASE – see 9.3.3 – services:

- the User_Information parameter of a COSEM-OPEN.request primitive, if present, is inserted into the "User data subfield" of the SNRM frame, sent during the data link connection establishment;
- if the SNRM frame received by the server contains a "User data subfield", its contents is passed to the server AP via the User_Information parameter of the COSEM-OPEN.indication primitive;

- the User_Information parameter of a COSEM-RELEASE.request primitive, if present, is inserted into the "User data subfield" of the DISC frame, sent during disconnecting the data link connection;
- if the DISC frame received by the server contains a "User data subfield", its contents is passed to the server AP via the User_Information parameter of the COSEM-RELEASE.indication primitive;
- the User_Information parameter of the COSEM-RELEASE.response primitive, if present, is inserted into the "User data subfield" of the UA or HDLC frame, sent in response to the DISC frame;
- if the UA or DM frame received by the client contains "User data subfield", its contents is passed to the client AP via the User_Information parameter of the COSEM-RELEASE.confirm primitive.

In addition, for the COSEM-ABORT .indication service primitive, the following rule applies:

- the Diagnostics parameter of the COSEM-ABORT.indication primitive – see 9.3.4 – may contain an unnumbered send status parameter. This parameter indicates whether, at the moment of the physical abort indication, the data link layer has or does not have a pending Unnumbered Information message (UI). The type and the value of this parameter is a local issue, thus it is not within the Scope of this Technical Report. See also the complete Green Book.

10.2.6.4 EventNotification service and protocol

This subclause describes the communication profile specific elements of the protocol of the EventNotification service.

In this profile, an event is reported always by the server Management Logical Device (mandatory, reserved upper HDLC address 0x01) to the Client Management AP (mandatory, reserved HDLC address 0x01).

The event-notification-request APDU is sent using connectionless data services, using an UI frame, at the first opportunity, i.e. when the server side data link layer receives the right to talk. The APDU shall fit into a single HDLC frame. To be able to send out the APDU, a physical connection between the physical device hosting the server and a client device must exist, and the server side data link layer needs to receive the token from the client side data link layer.

If there is a data link connection between the client and the server when the event occurs, the server side data link layer may send out the PDU – carrying the event-notification-request APDU – following the reception of an I, a UI or an RR frame from the client. See the complete Green Book.

Figure 45 shows the procedure in the case, when there is no physical connection when the event occurs (but this connection to a client device can be established).

NOTE Physical connection cannot be established when the server has only a local interface (for example an optical port as defined in IEC 62056-21) and the HHU, running the client application is not connected, or the server has a PSTN interface, but the telephone line is not available. Handling such cases is implementation specific.

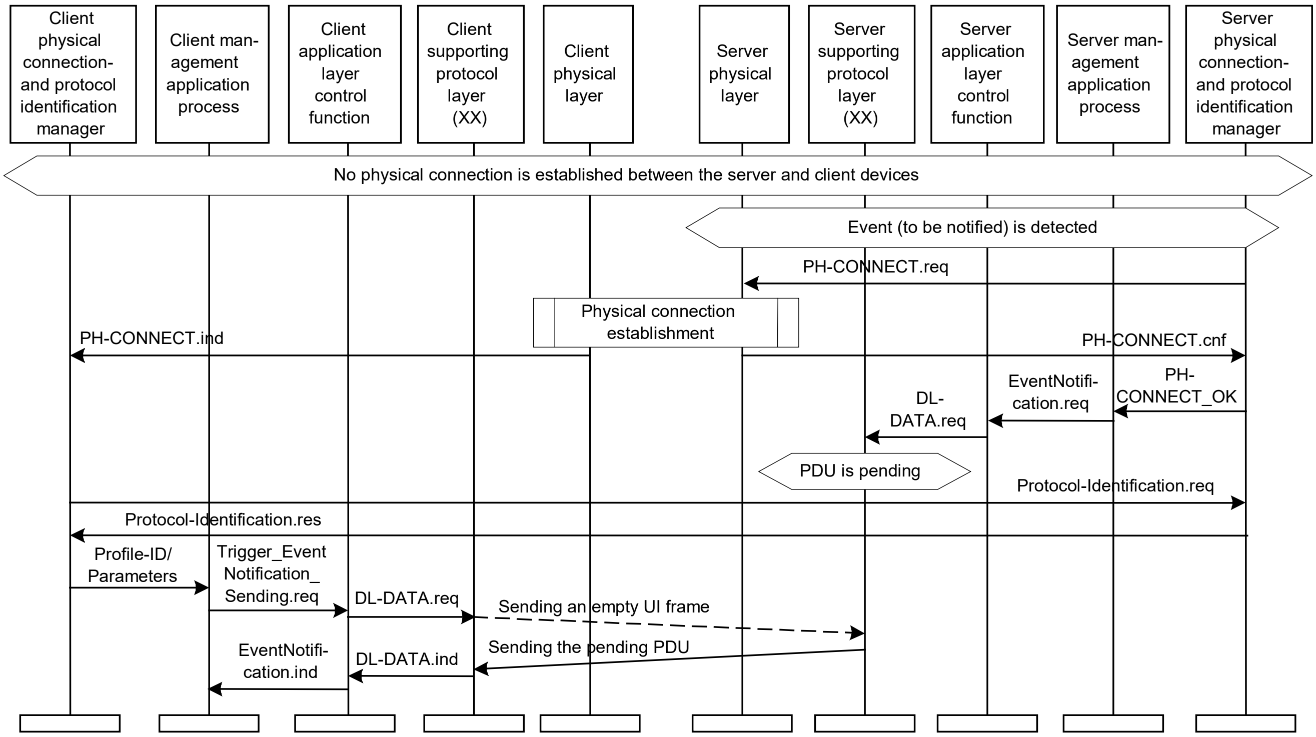

Client physical connection- and protocol identification manager
Client man-agement application process
Client application layer control function
Client supporting protocol layer (XX)
Client physical layer
Server physical layer
Server supporting protocol layer (XX)
Server application layer control function
Server man-agement application process
Server physical connection- and protocol identification manager
No physical connection is established between the server and client devices
Event (to be notified) is detected
PH-CONNECT.req
Physical connection establishment
PH-CONNECT.ind
PH-CONNECT.cnf
PH-CONNECT_OK
EventNotifi-cation.req
DL-DATA.req
PDU is pending
Protocol-Identification.req
Protocol-Identification.res
Profile-ID/ Parameters
Trigger_Event Notification_ Sending.req
DL-DATA.req
Sending an empty UI frame
Sending the pending PDU
EventNotifi-cation.ind
DL-DATA.ind

Figure 45 – Example: EventNotification triggered by the client

The first step is to establish this physical connection.

NOTE This physical connection establishment is done outside of the protocol stack.

If successful, this is reported at both sides to the physical connection manager process. At the server side, this indicates to the AP that the EventNotification.request service can be invoked now. When it is done, the server AL builds an event-notification-request APDU and invokes the connectionless DL-DATA.request primitive of the data link layer with the data parameter carrying the APDU. However, the data link layer may not be able to send this APDU, thus it is stored in the data link layer, waiting to be sent (pending).

When the client detects a successful physical connection establishment – and as there is no other reason to receive an incoming call – it supposes that this call is originated by a server intending to send the event-notification-request APDU.

At this moment, the client may not know the protocol stack used by the calling server. Therefore, it has to identify it first using the optional protocol identification service described in Clause 5. This is shown as a “Protocol-Identification.request” – “Protocol-Identification.response” message exchange in Figure 45. Following this, the client is able to instantiate the right protocol stack.

The client AP invokes then the TriggerEventNotificationSending .request primitive (see 9.3.12). Upon invocation of this primitive, the AL invokes the connectionless DL-DATA.request primitive of the data link layer with empty data, and the data link layer sends out an empty UI frame with the P/F bit set to TRUE, giving the permission to the server side data link layer to send the pending PDU.

When the client AL receives an event-notification-request APDU, it generates the EventNotification .indication primitive. The client is notified now about the event, the sequence is completed.

10.2.6.5 Transporting long messages

In this profile, the data link layer provides a method for transporting long messages in a transparent manner for the AL. This is described in the complete Green Book. See also 9.1.4.4.5.

As transparent long data transfer is specified only for the direction from the server to the client, the server side supporting protocol layer provides special services for this purpose to the server AL. As these services are specific to the supporting protocol layer, no specific AL services and protocols are specified for this purpose. When the supporting protocol layer supports transparent long data transfer, the server side AL implementation may be able to manage these services.

10.2.6.6 Supporting multi-drop configurations

A multi-drop arrangement is often used allowing a data collection system to exchange data with multiple physical metering equipment, using a shared communication resource like a telephone modem. Various physical arrangements are available, like a star, daisy chain or a bus topology. These arrangements can be modelled with a logical bus, to which the metering equipment and the shared resource are connected, see Figure 46.

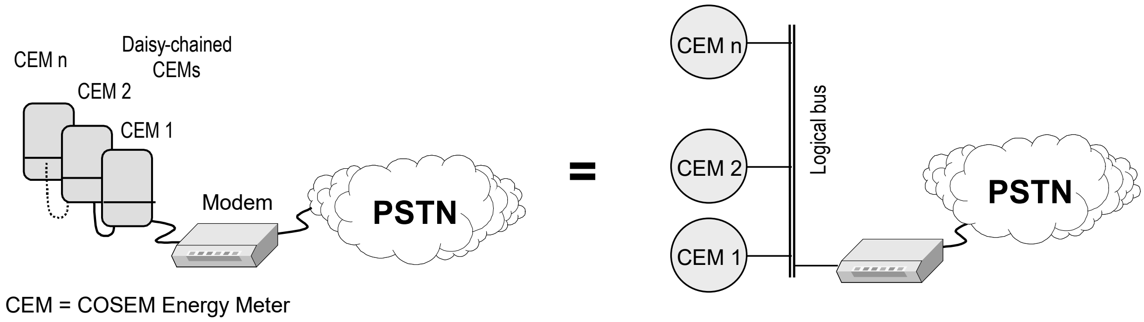


Figure 46 – Multi-drop configuration and its model

As collision on the bus must be avoided, but a protocol controlling access to the shared resource is not available, access to the bus must be controlled by external rules. In most cases, a Master-Slave arrangement is used, where the metering equipment are the Slaves. Slave devices have no right to send messages without first receiving an explicit permission from the Master.

In DLMS/COSEM, data exchange takes place based on the client/server model. Physical devices are modelled as a set of logical devices, acting as servers, providing responses to requests. Obviously, the Master Station of a multi-drop configuration is located at the other end of the communication channel and it acts as the client, sending requests and expecting responses.

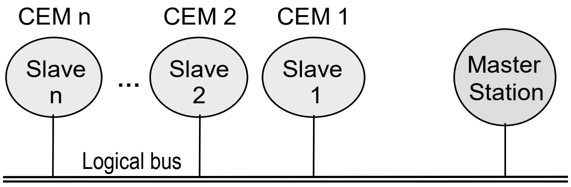


Figure 47 – Master/ Slave operation on the multi-drop bus

The client may send requests at the same time to multiple servers, if no response is expected (multicast or broadcast). If the client expects a response, it must send the request to a single server, giving also the right to talk. It has to wait then for the response before it may send a request to another server and with this, giving the right to talk. Arbitration of access to the common bus is thus controlled in a time-multiplexing fashion.

Messages from the client to the servers must contain addressing information. In this profile, it is ensured by using HDLC addresses. If a multi-drop arrangement is used, the HDLC address is split to two parts: the lower HDLC address to address physical devices and the upper HDLC address to address logical devices within the physical device. Both the lower and the upper address may contain a broadcast address. For details, see 8.4.2.

To be able reporting events, a server may initiate a connection to the client, using the unsolicited EventNotification / InformationReport services. As events in several or all servers connected to a multidrop may occur simultaneously – for example in the case of a power failure – they may initiate a call to the client simultaneously. For such cases, two problems have to be handled:

- collision on the logical bus: For the reasons explained above, several physical devices may try to access the shared resource (for example sending AT commands to the modem) simultaneously. Such situations must be handled by the manufacturers;
- identification of the originator of the event report: this is possible by using the CALLING Physical Device Address, as described in the complete Green Book.

10.3 The TCP-UDP/IP based communication profiles (COSEM_on_IP)

10.3.1 Targeted communication environments

The TCP-UDP/IP based communication profiles are suitable for remote data exchange with metering equipment via IP enabled networks such as Wide Area Networks, Neighbourhood Networks or Local Networks. This is shown in Figure 48.

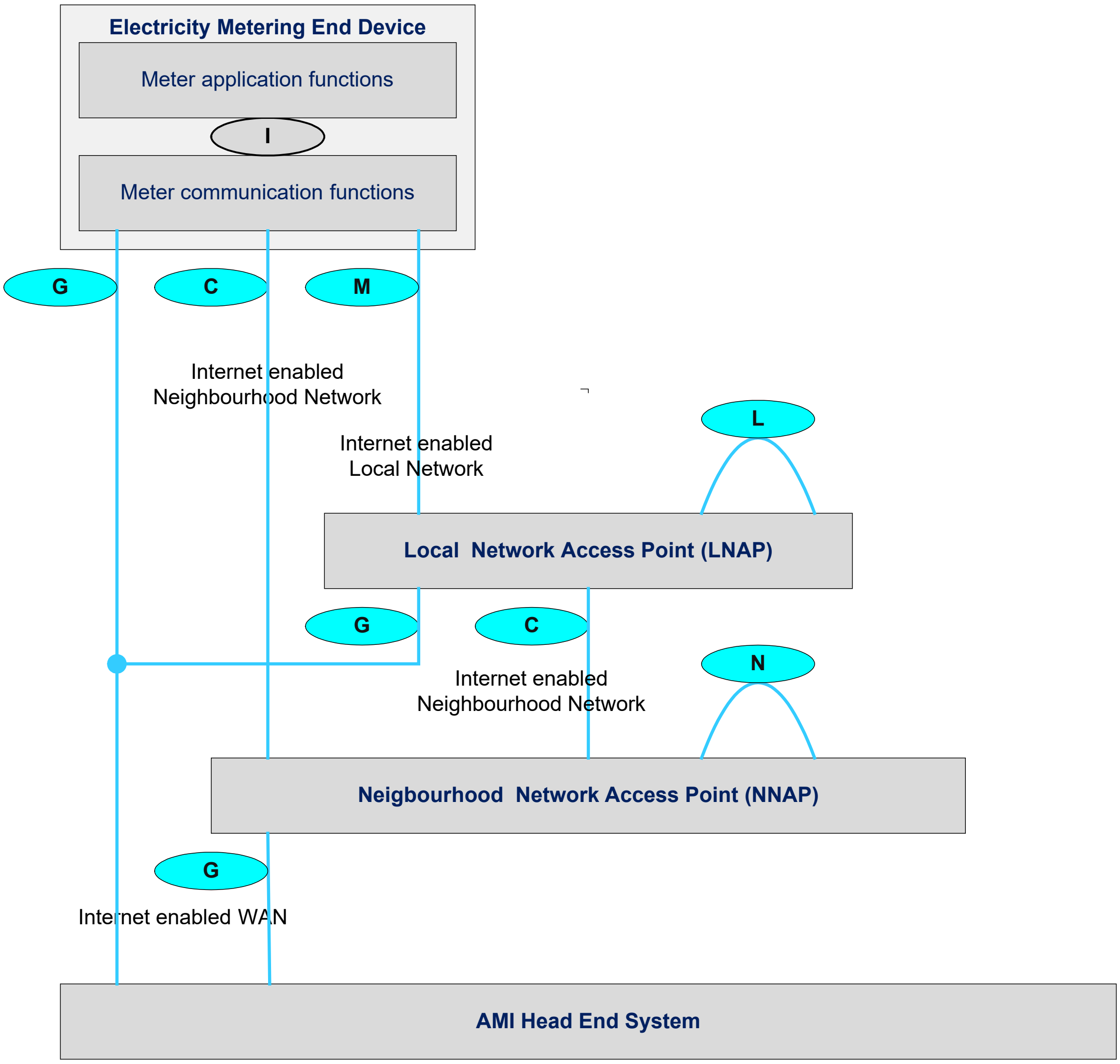

Electricity Metering End Device
Meter application functions
I
Meter communication functions
G
C
M
Internet enabled Neighbourhood Network
Internet enabled Local Network
L
Local Network Access Point (LNAP)
G
C
Internet enabled Neighbourhood Network
N
Neigbourhood Network Access Point (NNAP)
G
Internet enabled WAN
AMI Head End System

Figure 48 – Communication architecture

10.3.2 The structure of the profile(s)

The COSEM TCP-UDP/IP based communication profiles consist of five protocol layers:

- the DLMS/COSEM Application layer, specified in Clause 9;
- the DLMS/COSEM transport layer, specified in Clause 7;
- a network layer: the Internet Protocol (IPv4 or IPv6);
- a data link layer: any data link protocol supporting the network layer;
- a physical layer: any PhL supported by the data link layer chosen.

The DLMS/COSEM AL uses the services of one of the TLs (TCP or UDP) via a wrapper, which, in their turn, use the services of the IPv4 or IPv6 network layer to communicate with other nodes connected to this abstract network. The DLMS/COSEM AL in this environment can be considered as another Internet standard application protocol, which may co-exist with other Internet application protocols, like FTP, HTTP etc. See Figure 14.

The TCP-UDP/IP layers are implemented on a wide variety of real networks, which, just with the help of this IP Network abstraction, can be seamlessly interconnected to form Intra- and Internets using any set of lower layers supporting the Internet Protocol.

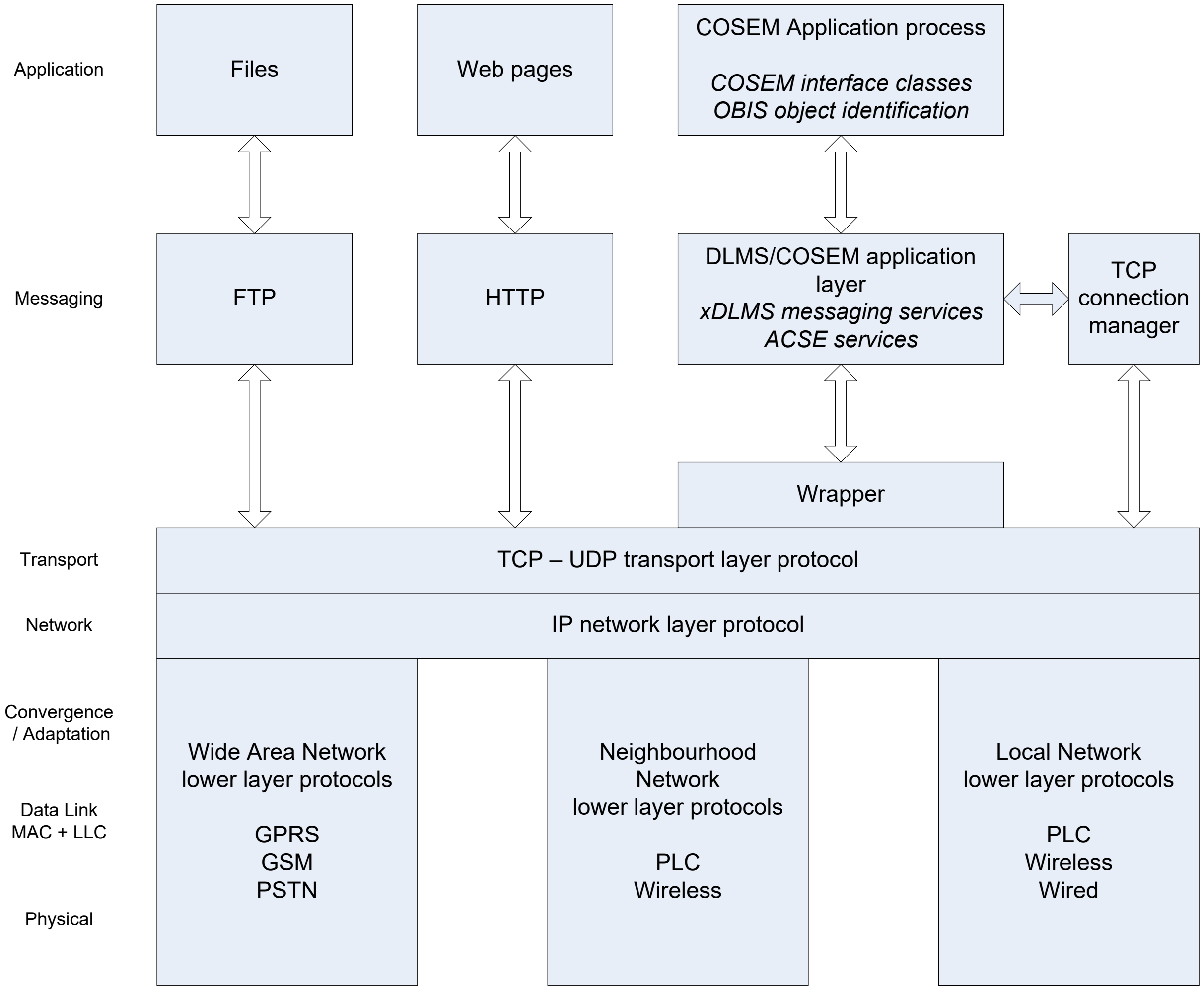

Application
Files
Web pages
COSEM Application process
COSEM interface classes
OBIS object identification
Messaging
FTP
HTTP
DLMS/COSEM application layer
xDLMS messaging services
ACSE services
TCP connection manager
Wrapper
Transport
TCP – UDP transport layer protocol
Network
IP network layer protocol
Convergence / Adaptation
Data Link MAC + LLC
Physical
Wide Area Network lower layer protocols
GPRS
GSM
PSTN
Neighbourhood Network lower layer protocols
PLC
Wireless
Local Network lower layer protocols
PLC
Wireless
Wired

Figure 49 – Examples for lower-layer protocols in the TCP-UDP/IP based profile(s)

Below the IP layer, a range of lower layers can be used. One of the reasons of the success of the Internet protocols is just their federating force. Practically any data networks, including Wide Area Networks such as GPRS, ISDN, ATM and Frame Relay, circuit switched PSTN and GSM networks (dial-up IP), Local Area Networks, such as Ethernet, neighbourhood networks and local networks using power line carrier or wireless protocols, etc. support TCP-UDP/IP networking.

Figure 49 shows a set of examples – far from being complete – for such communication networks and for the lower layer protocols used in these networks. Using the TCP-UDP/IP profile, DLMS/COSEM can be used practically on any existing communication network.

For more information see the complete Green Book.

11 AARQ and AARE encoding examples

11.1 General

This Clause 11 contains examples of encoding the AARQ and AARE APDUs, in cases of using various levels of authentication and in cases of success and failure.

The AARQ, AARE, RLRQ and RLRE APDUs – see 9.4.3.1 – shall be encoded in BER (ISO/IEC 8825-1:2015). The user-information field of the AARQ and AARE APDUs contains the xDLMS InitiateRequest / InitiateResponse or confirmedServiceError APDUs respectively, encoded in A-XDR as OCTET STRING.

11.2 Encoding of the xDLMS InitiateRequest / InitiateResponse APDU

The xDLMS InitiateRequest / InitiateResponse APDUs are specified as follows:

```
InitiateRequest ::= SEQUENCE
{
--  shall not be encoded in DLMS without ciphering
    dedicated-key                           OCTET STRING OPTIONAL,
    response-allowed                        BOOLEAN DEFAULT TRUE,
    proposed-quality-of-service             IMPLICIT Integer8 OPTIONAL,
    proposed-dlms-version-number            Unsigned8,
    proposed-conformance                    Conformance,
    client-max-receive-pdu-size             Unsigned16
}

InitiateResponse ::= SEQUENCE
{
    negotiated-quality-of-service           IMPLICIT Integer8 OPTIONAL,
    negotiated-dlms-version-number          Unsigned8,
    negotiated-conformance                  Conformance,
    server-max-receive-pdu-size             Unsigned16,
    vaa-name                                ObjectName
}
```

The xDLMS InitiateRequest and InitiateResponse APDUs are encoded in A-XDR and they are inserted in the user-information field of the AARQ / AARE APDU respectively.

In the examples below, the following values are used:

- dedicated key: not present; no ciphering is used;
- response-allowed: TRUE (default value);
- proposed-quality-of-service and negotiated-quality-of-service: not present (not used in DLMS/COSEM);
- proposed-conformance and negotiated-conformance: see below;
- proposed-dlms-version-number and negotiated-dlms-version-number = 6;
- client-max-receive-pdu-size: 1200_D = 0x04B0;
- server-max-receive-pdu-size: 500_D = 0x01F4;
- vaa-name in the case of LN referencing: the dummy value 0x0007;
- vaa-name in the case of SN referencing: the base_name of the current Association SN object, 0xFA00.
- The proposed-conformance and the negotiated-conformance elements carry the proposed conformance block and the negotiated conformance block respectively. The values of these examples, for LN referencing and SN referencing respectively, are shown in Table 16.

Table 16 – Conformance block

Conformance ::= **[APPLICATION 31] IMPLICIT BIT STRING** (SIZE(24))		LN referencing		SN referencing	
-- the bit is set when the corresponding service or functionality is available	**Used with**	**Proposed**	**Negotiated**	**Proposed**	**Negotiated**
reserved-zero (0),		0	0	0	0
reserved-one (1),		0	0	0	0
reserved-two (2),		0	0	0	0
read (3),	SN	0	0	1	1
write (4),	SN	0	0	1	1
unconfirmed-write (5),	SN	0	0	1	1
reserved-six (6),		0	0	0	0
reserved-seven (7),		0	0	0	0
attribute0-supported-with-set (8),	LN	0	0	0	0
priority-mgmt-supported (9),	LN	1	1	0	0
attribute0-supported-with-get (10),	LN	1	0	0	0
block-transfer-with-get-or-read (11),	LN	1	1	0	0
block-transfer-with-set-or-write(12),	LN	1	0	0	0
block-transfer-with-action (13),	LN	1	0	0	0
multiple-references (14),	LN / SN	1	0	1	1
information-report (15),	SN	0	0	1	1
reserved-sixteen (16),		0	0	0	0
reserved-seventeen (17),		0	0	0	0
parameterized-access (18),	SN	0	0	1	1
get (19),	LN	1	1	0	0
set (20),	LN	1	1	0	0
selective-access (21),	LN	1	1	0	0
event-notification (22),	LN	1	1	0	0
action (23)	LN	1	1	0	0
Value of the bit string		00 7E 1F	00 50 1F	1C 03 20	1C 03 20

With these parameters, the A-XDR encoding of the xDLMS InitiateRequest APDU is as shown in Table 17:

Table 17 – A-XDR encoding the xDLMS InitiateRequest APDU

-- A-XDR encoding the xDLMS InitiateRequest APDU	**LN referencing**	**SN referencing**
// encoding of the tag of the xDLMS APDU CHOICE *(InitiateRequest)*	01	01
*-- encoding of the dedicated-key component (**OCTET STRING OPTIONAL**)*		
// usage flag*(**FALSE**, not present)*	00	00
-- encoding of the response-allowed component *(**BOOLEAN DEFAULT TRUE**)*		
// usage flag*(**FALSE**, default value **TRUE** conveyed)*	00	00
*-- encoding of the proposed-quality-of-service component ([0] **IMPLICIT** Integer8 **OPTIONAL**)*		
// usage flag*(**FALSE**, not present)*	00	00
-- encoding of the proposed-dlms-version-number component (Unsigned8)		
// value= 6, the encoding of an Unsigned8 is its value	06	06
*-- encoding of the proposed-conformance component (Conformance, [APPLICATION 31] **IMPLICIT BIT STRING** (SIZE(24))* [1]		
// encoding of the [APPLICATION 31] tag *(ASN.1 explicit tag)* [2]	5F 1F	5F 1F
// encoding of the length of the 'contents' field in octet *(4)*	04	04
// encoding of the number of unused bits in the final octet of the BIT STRING *(0)*	00	00
// encoding of the fixed length BIT STRING value	00 7E 1F	1C 03 20
-- encoding of the client-max-receive-pdu-size component (Unsigned16)		
// value = 0x04B0, the encoding of an Unsigned16 is its value	04 B0	04 B0
-- resulting octet-string, to be inserted in the user-information field of the AARQ APDU	01 00 00 00 06 5F 1F 04 00 00 7E 1F 04 B0	01 00 00 00 06 5F 1F 04 00 1C 03 20 04 B0

[1] As specified in IEC 61334-6, Annex C, Examples 1 and 2, the proposed-conformance element of the xDLMS InitiateRequest APDU and the negotiated-conformance element of the xDLMS InitiateResponse APDU are encoded in BER. That's why the length of the bit-string and the number of the unused bits are encoded.

[2] For encoding of identifier octets, see ISO/IEC 8825-1:2015, 8.1.2. For compliance with existing implementations, encoding of the [Application 31] tag on one byte (5F) instead of two bytes (5F 1F) is accepted when the 3-layer, connection-oriented, HDLC based profile is used.

The A-XDR encoding of the xDLMS InitiateResponse APDU is as shown in Table 18.

Table 18 – A-XDR encoding the xDLMS InitiateResponse APDU

-- A-XDR encoding the xDLMS InitiateResponse APDU	**LN referencing**	**SN referencing**
// encoding of the tag of the xDLMS APDU CHOICE *(InitiateResponse)*	08	08
-- encoding of the negotiated-quality-of-service component ([0] ***IMPLICIT*** *Integer8* ***OPTIONAL****)*		
// usage flag*(**FALSE**, not present)*	00	00
-- encoding of the negotiated-dlms-version-number component (Unsigned8)		
// value = 6, the encoding of an Unsigned8 is its value	06	06
-- encoding of the negotiated-conformance component (Conformance, [APPLICATION 31] ***IMPLICIT BIT STRING*** *(SIZE(24))*		
// encoding of the [APPLICATION 31] tag *(ASN.1 explicit tag)*	5F 1F	5F 1F
// encoding of the length of the 'contents' field in octet *(4)*	04	04
// encoding of the number of unused bits in the final octet of the BIT STRING *(0)*	00	00
// encoding of the fixed length BIT STRING value	00 50 1F	1C 03 20
-- encoding of the server-max-receive-pdu-size component (Unsigned16)		
// value = 0x01F4, the encoding of an Unsigned16 is its value	01 F4	01 F4
-- encoding of the VAA-Name component (ObjectName, Integer16)		
// value=0x0007 for LN and 0xFA00 for SN referencing; the encoding of a value constrained Integer16 is its value	00 07	FA 00
-- resulting octet-string, to be inserted in the user-information field of the AARE APDU	08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07	08 00 06 5F 1F 04 00 1C 03 20 01 F4 FA 00

For more information see the complete Green Book.

12 Encoding examples: AARQ and AARE APDUs using a ciphered application context

12.1 A-XDR encoding of the xDLMS InitiateRequest APDU, carrying a dedicated key

NOTE The System Title is the same in each example. In reality, the System Title in the request and in the response APDUs should be different, as they are originated by different systems.

In this example:

- the value of the dedicated key is 00112233445566778899AABBCCDDEEFF;
- the value of the Conformance block is 007E1F;
- the value of the client-max-receive-pdu-size is 1 200 bytes (0x04B0).

The A-XDR encoding of the xDLMS InitiateRequest APDU carrying a dedicated key is shown in Table 19.

Table 19 – A-XDR encoding of the xDLMS InitiateRequest APDU

// encoding of the tag of the xDLMS APDU CHOICE *(InitiateRequest)*	01
-- encoding of the dedicated-key component (**OCTET STRING OPTIONAL**)	
// usage flag (**TRUE**, present)	01
// length of the **OCTET STRING**	10
// contents of the **OCTET STRING**	0011223344556677 8899AABBCCDDEEFF
*-- encoding of the response-allowed component (**BOOLEAN DEFAULT TRUE**)*	
// usage flag *(**FALSE**, default value **TRUE** conveyed)*	00
-- encoding of the proposed-quality-of-service component *([0] **IMPLICIT** Integer8 **OPTIONAL**)*	
// usage flag (**FALSE**, not present)	00
-- encoding of the proposed-dlms-version-number component (Unsigned8)	
// value = 6; the A-XDR encoding of an Unsigned8 is its value	06
*-- encoding of the proposed-conformance component (Conformance, [APPLICATION 31] **IMPLICIT BIT STRING** (SIZE(24))* [1]	
// encoding of the [APPLICATION 31] tag *(ASN.1 explicit tag)* [2]	5F1F
// encoding of the length of the 'contents' field in octet *(4)*	04
// encoding of the number of unused bits in the final octet of the BIT STRING *(0)*	00
// encoding of the fixed length BIT STRING value	007E1F
-- encoding of the client-max-receive-pdu-size component (Unsigned16)	
// value = 0x04B0, the encoding of an Unsigned16 is its value	04B0
-- resulting octet-string	0101100011223344 5566778899AABBCC DDEEFF0000065F1F 0400007E1F04B0

[1] As specified in IEC 61334-6:2000, Annex C, Examples 1 and 2, the proposed-conformance element of the xDLMS InitiateRequest APDU and the negotiated-conformance element of the xDLMS InitiateResponse APDU are encoded in BER. That's why the length of the bit-string and the number of the unused bits are encoded.

[2] For encoding of identifier octets, see the complete Green Book. For compliance with existing implementations, encoding of the [Application 31] tag on one byte (5F) instead of two bytes (5F 1F) is accepted when the 3-layer, connection-oriented, HDLC based profile is used.

For more information see the complete Green Book.

13 S-FSK PLC encoding examples

13.1 CI-PDUs, ACSE APDUs and xDLMS APDUs carried by MAC frames using the IEC 61334-4-32 LLC sublayer

In these examples, the following communication sequence is shown, when the DLMS/COSEM S-FSK PLC profile is used with the IEC 61334-4-32:1996 LLC sublayer:

- the initiator Discovers, then Registers a new server system;
- the initiator establishes an AA;
- it reads the time attribute of the Clock object (once and 13 times, to show block transfer);
- the initiator Pings a server;
- the initiator sends a RepeaterCall service.

In these examples: SYSTEM-TITLE-SIZE = 6.

The traces have been taken from a protocol analyser. The contents of the MAC frame are explained. The MAC frame is shown between the brackets () following the “02 xx 50” header and followed by 00 00 (final field, normally a frame check). The Pad fields are not shown.

For more information see the complete Green Book.

14 Data transfer service examples

14.1 GET / Read, SET / Write examples

Tables within the complete Green Book show examples for data exchange using xDLMS services with LN referencing (left column) and SN referencing (right column). Table 20 shows the objects used in the examples.

Table 20 – The objects used in the examples

```
Object 1:
-   Class: Data
-   Logical name: 0000800000FF
-   Short name of value attribute: 0100
-   Value: octet string of 50 elements
-   01020304050607080910111213141516
-   17181920212223242526272829303132
-   33343536373839404142434445464748
-   4950

Object 2:
-   Class: Data
-   Logical name: 0000800100FF
-   Short name of value attribute: 0110
-   Value: visible string of 3 elements 303030
```

In the case of block transfer, the negotiated APDU size is 40 bytes.

Nota bene: What is negotiated is the APDU size not the block size! Therefore, the block size is smaller than the APDU size.

For more information see the complete Green Book.

14.2 ACCESS service example

Table 21 shows an example of the ACCESS service without general block transfer.

Table 21 – Example: ACCESS service without block transfer

Message Elements (MAX APDU = 1024)	Contents	LEN (Bytes)
Access-Request	D9	1
long-invoke-id-and-priority	40000000	4
date-time	00	1
access-request-body		0
access-request-specification		0
SEQUENCE OF CHOICE	04	1
access-request-get	01	1
cosem-attribute-descriptor		0
class-id	0001	2
instance-id	0000600100FF	6
attr-id	02	1
access-request-get	01	1
cosem-attribute-descriptor		0
class-id	0008	2
instance-id	0000010000FF	6
attr-id	02	1

access-request-set	02	1
cosem-attribute-descriptor		0
class-id	0014	2
instance-id	00000D0000FF	6
attr-id	07	1
access-request-set	02	1
cosem-attribute-descriptor		0
class-id	0014	2
instance-id	00000D0000FF	6
attr-id	08	1
access-request-list-of-data		
SEQUENCE OF Data	04	1
null-data	00	1
null-data	00	1
array	01040203090100090CFFFFFFFFFFFFFF 00000000000901FF0203090101090CFF FFFFFFFFFFFF00000000000901FF0203 090102090CFFFFFFFFFFFFFF00000000 000901FF0203090103090CFFFFFFFFFF FFFF00000000000901FF	90
array	01040208090100 11FF11FF11FF11FF11 FF11FF11FF020809010111021101110 1 1101110111011101020809010211FF11 FF11FF11FF11FF11FF11FF0208090103 1101110211021102110211021102	78
Complete Access-Request APDU (encoded)	D94000000000040100010000600100FF 020100080000010000FF020200140000 0D0000FF0702001400000D0000FF0804 000001040203090100090CFFFFFFFFFF FFFF00000000000901FF020309010109 0CFFFFFFFFFFFFFF00000000000901FF 020309010209 0CFFFFFFFFFFFFFF0000 000000 0901FF020309010309 0CFFFFFF FFFFFFFF00000000000901FF01040208 09010011FF11FF11FF11FF11FF11FF11 FF020809010111021101110111011101 11011101020809010211FF11FF11FF11 FF11FF11FF11FF020809010311011102 11021102110211021102	218
Access-Response	DA	1
long-invoke-id-and-priority	40000000	4
date-time	00	1
access-response-body		0
access-request-specification OPTIONAL	00	1
access-response-list-of-data		0
SEQUENCE OF Data	04	1
octet-string	09083030303030303031	10
octet-string	090C07DC030C07161E0000FF8880	14
null-data	00	1
null-data	00	1
access-response-specification		0
SEQUENCE OF CHOICE	04	1

access-response-get	01	1
result	00	1
access-response-get	01	1
result	00	1
access-response-set	02	1
result	00	1
access-response-set	02	1
result	00	1
Complete Access-Response APDU (encoded)	DA400000000000004090830303030303030 3031090C07DC030C07161E0000FF8880 0000040100010002000200	43

14.3 Compact array encoding example

14.3.1 General

Any series of data of the same type can be encoded as array or compact-array.

The compact-array data type is present from the beginning in the DLMS/COSEM specification, but so far its interpretation was not unambiguous, therefore it has not been used.

The objective of this subclause 14.3 is to facilitate the use of compact-array encoding.

For more information see the complete Green Book.

Bibliography

IEC 60050-300, *International Electrotechnical Vocabulary – Revision of Chapter 301, 302, 303 – Electrical measurements and measuring instruments - Chapter 311: General terms relating to measurements Chapter 312: General terms relating to electrical measurements - Chapter 313: Types of electrical measuring instrument - Chapter 314: Specific terms according to the type of instrument*

IEC 60870-5-1:1990, *Telecontrol equipment and systems. Part 5: Transmission protocols – Section One: Transmission frame formats*

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 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 TS 62056-1-1, Electricity metering data exchange – The DLMS/COSEM suite – Part 1-1: Template for DLMS/COSEM communication profile standards

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

IEC 62056-41:1998, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 41: Data exchange using wide area networks: Public switched telephone network (PSTN) with LINK+ protocol*

IEC 62056-42:2002, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 42: Physical layer services and procedures for connection-oriented asynchronous data exchange*

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

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

IEC 62056-4-7, ELECTRICITY METERING DATA EXCHANGE – The DLMS/COSEM suite – Part 4-7: DLMS/COSEM transport layer for IP networks
This standard cancels and replaces IEC 62056-47:2006.

IEC/TS 62056-51:1998, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 51: Application layer protocols*

IEC/TS 62056-52:1998, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 52: Communication protocols management distribution line message specification (DLMS) server*

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

IEC 62056-6-1, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-1: Object Identification System (OBIS)*

IEC 62056-6-2, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes*

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

IEC 62056-7-6:2013, *ELECTRICITY METERING DATA EXCHANGE – The DLMS/COSEM suite – Part 7-6: The 3-layer, connection-oriented HDLC based communication profile*

IEC 62056-8-3, *ELECTRICITY METERING DATA EXCHANGE – The DLMS/COSEM suite – Part 8-3: PLC S-FSK communication profile for neighbourhood networks*

IEC 62056-8-4, *ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 8-4: Narrow-band OFDM PRIME PLC communication profile for neighbourhood networks*

IEC 62056-8-5, *ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 8-5: Narrow-band OFDM G3-PLC communication profile for neighbourhood networks*

IEC 62056-9-7, *ELECTRICITY METERING DATA EXCHANGE – The DLMS/COSEM suite – Part 9-7: Communication profile for TCP-UDP/IP networks*

ISO/IEC 9545:1994, *Information technology - Open Systems Interconnection – Application layer structure*

Evaluation of ISO/IEC 9798 Protocols Version 2.0 David Basin and Cas Cremers April 7, 2011

ISO/IEC 9798-1:2010, *Information technology — Security techniques — Entity authentication — Part 1: General*

ISO/IEC 9798-2 Ed. 3:2008, *Information technology — Security techniques — Entity authentication — Part 2: Mechanisms using symmetric encipherment algorithms*

ISO/IEC 9798-3:1998, *Information technology – Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques*

ISO/IEC 10731:1994, *Information technology - Open Systems Interconnection - Basic Reference Model - Conventions for the definition of OSI services*

ISO/IEC 15945:2002, *Information technology — Security techniques — Specification of TTP services to support the application of digital signatures*

ISO 2110:1989, *Information technology – Data communication – 25-pole DTE/DCE interface connector and contact number assignments*
ITU-T V.24:1996, *List of definitions for interchange circuits between data terminal equipment (DTE) and data circuit-terminating equipment (DCE)*

ITU-T V.25:1996, *Automatic answering equipment and general procedures for automatic calling equipment on the general switched telephone network*

ITU-T V.25bis:1996, *Synchronous and asynchronous automatic dialling procedures on switched networks*

ITU-T V.28:1993, *Electrical characteristics for unbalanced double-current interchange circuits*

ITU-T X.811:1995, *Information technology - Open Systems Interconnection - Security frameworks for open systems: Authentication framework*

IEEE 802.1 ae:2006, *IEEE Standard for Local and metropolitan area networks Media Access Control (MAC) Security*

IEEE 802.15.4:2006, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications*

CEN/CLC/ETSI TR 50572 *Functional reference architecture for communications in smart metering systems*

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 800-47:2002, *Security Guide for Interconnecting Information Technology Systems*

The Galois/Counter Mode of Operation (GCM) - David A. McGrew, Cisco Systems, Inc. 170, West Tasman Drive, San Jose, CA 95032, mcgrew@cisco.com, John Viega, Secure Software, 4100 Lafayette Center Drive, Suite 100, Chantilly, VA 20151, viega@securesoftware.com, May 31, 2005

RFC 0791, *Internet Protocol, 1981, Also: STD0005, Updated by: RFC 1349, Obsoletes: RFC 0760,* *http://tools.ietf.org/html/rfc791*

RFC 0792, *Internet Control Message Protocol, 1981, Also: STD0005, Updated by: RFC 0950, Obsoletes: RFC0777,* *http://tools.ietf.org/html/rfc792*

RFC 0793, *Transmission Control Protocol, 1981, Also: STD0007, Updated by: RFC 3168,* *http://tools.ietf.org/html/rfc793*

RFC 0822, Standard for the format of ARPA Internet Text Messages, 1982, *http://www.ietf.org/rfc/rfc822*

RFC 0826, *Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, 1982, Also: STD0037,* *http://tools.ietf.org/html/rfc826*

RFC 0894, *Standard for the transmission of IP datagrams over Ethernet networks, 1984, Also: STD0041,* *http://tools.ietf.org/html/rfc894*

RFC 0919, *Broadcasting Internet Datagrams, 1984, Also: STD0005,* *http://tools.ietf.org/html/rfc919*

RFC 0922, *Broadcasting Internet datagrams in the presence of subnets,1984, Also: STD0005,* *http://tools.ietf.org/html/rfc922*

RFC 0950, *Internet Standard Subnetting Procedure, 1985, Also: STD0005, Updates: RFC 0792,* http://tools.ietf.org/html/rfc950

RFC 1042, *Standard for the transmission of IP datagrams over IEEE 802 networks, 1988, Also: STD0043, Obsoletes: RFC0948,* *http://www.ietf.org/rfc/rfc1042.txt*

RFC 1095, *The Common Management Information Services and Protocol over TCP/IP, (CMOT), 1989,* *http://www.ietf.org/rfc/rfc1095*

RFC 1112,*Host extensions for IP multicasting, 1989, Also: STD0005, Updated by: RFC 2236, Obsoletes: RFC0988, RFC1054,* *https://tools.ietf.org/rfc/rfc1112.txt*

RFC 1321, *The MD5 Message-Digest Algorithm, 1982,* *http://www.ietf.org/rfc/rfc1321.txt*

RFC 1662, *PPP in HDLC-like Framing, 1984,* *http://www.ietf.org/rfc/rfc1662.txt*

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication, 2004,* *http://www.ietf.org/rfc/rfc2104.txt*

RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels, 1997,* *http://www.ietf.org/rfc/rfc2119.txt*

RFC 2315, *PKCS #7, Cryptographic Message Syntax Version 1.5, 1998,* https://www.ietf.org/rfc/rfc2315

RFC 2560 X.509, *Internet Public Key Infrastructure – Online Certificate Status Protocol – OCSP, 1999,* http://www.ietf.org/rfc/rfc2560

RFC 2822, *Internet Message Format, 2001,* http://www.ietf.org/rfc/rfc2822

RFC 2986, *PKCS #10 v1.7: Certification Request Syntax Standard,* http://www.ietf.org/rfc/rfc2986

RFC 3268, *Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS), 2002,* http://tools.ietf.org/html/rfc3268

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP), 2005,* https://www.rfc-editor.org/rfc/rfc4106.txt

RFC 4210, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP), 2005,* http://www.ietf.org/rfc/rfc4210.txt

RFC 4211, *Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF), 2005,* http://www.ietf.org/rfc/rfc4211

RFC 4308, *Cryptographic Suites for IPsec, 2005,* https://www.google.hu/#q=RFC%204308

RFC 4835, *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH),2007,* http://tools.ietf.org/html/rfc4335

RFC 5084, *Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS), 2007,* https://www.google.hu/#q=RFC+5084

RFC 5349, *Elliptic Curve Cryptography (ECC) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT), 2008,* https://tools.ietf.org/html/rfc5349

RFC 5480, *Elliptic Curve Cryptography Subject Public Key Information, 2009,* http://www.ietf.org/rfc/rfc5480.txt

RFC 5758, *Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA, 2010,* http://www.ietf.org/rfc/rfc5758.txt

RFC 5759, *Suite B Certificate and Certificate Revocation List (CRL) Profile, 2010,* http://tools.ietf.org/search/rfc5759

RFC 6024, *Trust Anchor Management Requirements* http://www.ietf.org/rfc/rfc6024.txt

RFC 6318, *Suite B in Secure/Multipurpose Internet Mail Extensions (S/MIME), 2011,* http://tools.ietf.org/html/rfc6318

SEC1:2009, *Standards for Efficient Cryptography: Elliptic Curve Cryptography. SECG. Version 2.0*

SEC2:2010, *Standards for Efficient Cryptography: Recommended Elliptic Curve Domain Parameters, Version 2.0. Certicom Research*

UK DECC Smart Metering Implementation Programme – Great Britain Companion Specification (GBCS) V0.7 Rev6

Technische Richtlinie BSI TR-03109-4: Public Key Infrastruktur für Smart Meter Gateways Version 1 – 18.03.2013. Publicly available at:

https://www.bsi.bund.de/DE/Themen/SmartMeter/TechnRichtlinie/TR_node.html