EXCERPT FROM

Companion Specification
for Energy Metering

DLMS/COSEM

Architecture
and Protocols

DLMS User Association

dlms™
device
language
message
specification

Table of Contents

Figures

Tables

1. Foreword

Copyright

Intellectual Property Rights

Attention is drawn to the fact that the RepeaterCall service specified in this document is covered by the patent EP 0 638 886 B1. The holder of the patent, Landis+Gyr, has assured the DLMS UA that it is prepared to grant a license to an unrestricted number of applicants on a worldwide, non-discriminatory basis and on reasonable terms and conditions to make, use and sell implementations of the above document.

Acknowledgement

The actual document has been established by a team of experts working for members of the DLMS User Association.

Status of standardization

The contents of this edition will be used to prepare a revision of IEC 62056-53, COSEM Application layer.

2. 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 buliding blocks to model this functionality. The model does not cover internal, implementation-specific issues.

Communication protocols define how the data can 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.

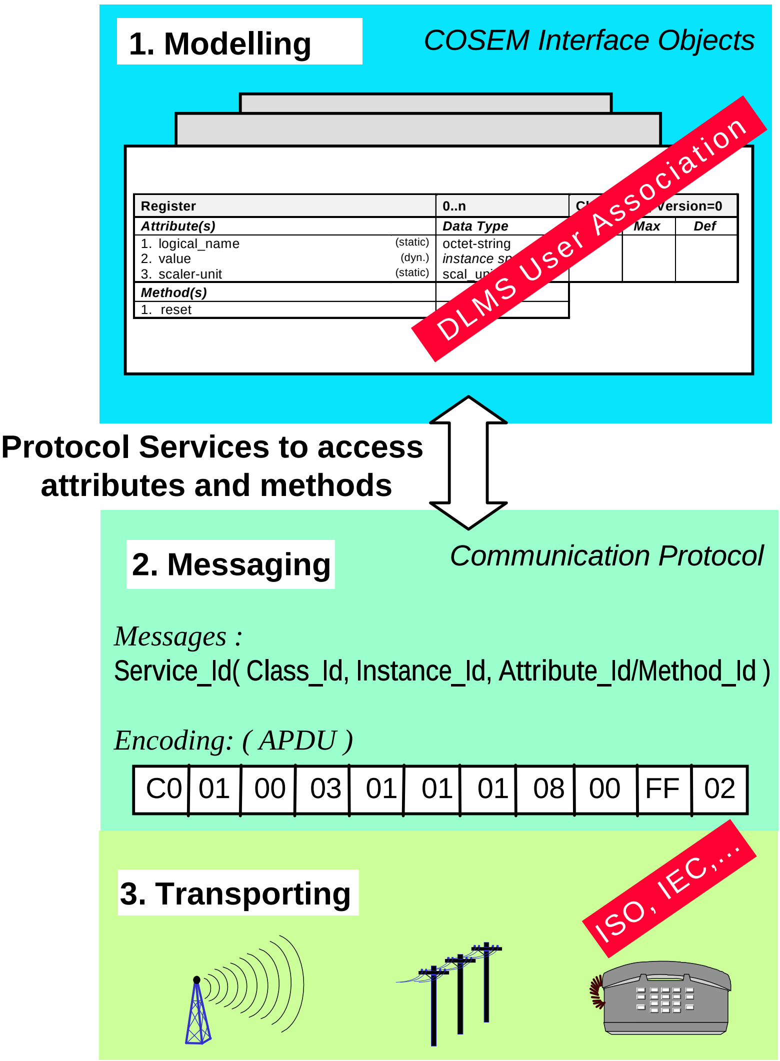


Figure 1 – The three steps approach of COSEM: Modelling – Messaging – Transporting

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 document. It specifies communication profiles for various communication media and the protocol layers of these communication profiles. The top layer in any profile is the COSEM application layer. It provides a logical connection between the client and the server(s). It also provides the xDLMS messaging services to access attributes and methods of the COSEM interface objects. The lower, communication profile specific protocol layers transport the information.

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

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

3. Introduction

3.1 Referenced documents

Ref.	Title
DLMS UA 1000-1 Ed. 9.0:2009	*COSEM Interface Classes and the OBIS Identification System, "Blue Book"*
DLMS UA 1001-1 Ed. 3.0:2007	*DLMS/COSEM Conformance Testing Process, "Yellow Book"*
DLMS UA 1002 Ed. 1.0:2003	*COSEM Glossary of Terms, "White Book"*
IEC 61334-4-1 Ed. 1.0:1996	*Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 1: Reference model of the communication system*
IEC 61334-4-32 Ed. 1.0:1996	*Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 32: Data link layer – Logical link control (LLC)*
IEC 61334-4-41 Ed.1.0:1996	*Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 41: Application protocol – Distribution line message specification*
IEC 61334-4-511:2000	*Distribution automation using distribution line carrier systems – Part 4-511: Data communication protocols – Systems management – CIASE protocol*
IEC 61334-4-512 Ed. 1.0:2001	*Distribution automation using distribution line carrier systems – Part 4-512: Data communication protocols – System management using profile 61334-5-1 – Management Information Base (MIB)*
IEC 61334-5-1 Ed. 2.0:2001	*Distribution automation using distribution line carrier systems – Part 5-1: Lower layer profiles – The spread frequency shift keying (S-FSK) profile*
IEC 61334-6 Ed 1.0:2000	*Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule*
IEC 62056-21 Ed 1.0:2002	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange*
IEC 62056-31 Ed 1.0 :1999	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 31: Use of local area networks on twisted pair with carrier signalling*
IEC 62056-42 Ed.1.0:2002	*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 Ed.1.1:2007	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol*
IEC 62056-47 Ed 1.0:2006	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 47: COSEM transport layer for IP networks*
IEC 62056-53 Ed 2.0:2006	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 53: COSEM Application layer*
IEC 62056-61 Ed 2.0:2006	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 61: OBIS Object identification system*
IEC 62056-62 Ed 2.0:2006	*Electricity metering – Data exchange for meter reading, tariff and load control – Part 62: Interface classes*
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*
ISO/IEC 8650-1 Ed 2.0:1996	*Information technology – Open systems interconnection – Connection-oriented protocol for the association control service element: Protocol specification*
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 Ed. 3:2002	*Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

Ref.	Title
ISO/IEC 8825 Ed. 3:2002	*Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*
ISO/IEC 13239 Ed. 3.0:2002	*Information Technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures*
EN 13757-2 Ed. 1.0:2004	*Communication system for and remote reading of meters – Part 2 : Physical and Link Layer, Twisted Pair Baseband (M-Bus)*
ANSI C12.21:1999	*Protocol Specification for Telephone Modem Communication*
FIPS PUB 180-1:2002	*Secure hash standard*
FIPS PUB 197:2001,	*Advanced Encryption Standard (AES)*
FIPS PUB 198:2002	*The Keyed-Hash Message Authentication Code (HMAC)*
FIPS PUB 199:2002	*Standards for Security Categorization of Federal Information and Information Systems*
NIST SP 800-21:2005	*Guideline for Implementing Cryptography in the Federal Government*
NIST SP 800-38D:2007	*Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
NIST SP 800-57:2006	*Recommendation for Key Management – Part 1: General (Revised)*
RFC 1321:1992	*The MD5 Message-Digest Algorithm*
RFC 3394:2002	*Advanced Encryption Standard (AES) Key Wrap Algorithm*
RFC 5084:2007	*Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)*
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*

3.2 Abbreviations

Abbreviation	Explanation
AA	Application Association
AAD	Additional Authenticated Data (used with ciphering of APDUs)
AARE	A-Associate Response – an APDU of the ACSE
AARQ	A-Associate Request – an APDU of the ACSE
ACPM	Association Control Protocol Machine
ACSE	Assoication 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
ARP	Address Resolution Protocol
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
CF	Control Function
CL	Connectionless
class_id	Class identification code
client	A station, asking for services. Normally the master station
.cnf	.confirm service primitive
CO	Connection-oriented
COSEM	Companion Specification for Energy Metering
COSEM Interface Object	An instance of a COSEM Interface Class
COSEM_on_IP	The TCP-UDP/IP based COSEM communication profile
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)
DPDU	Data Link Protocol Data Unit
DSAP	Data Link Service Access Point
DSDU	Data Link Service Data Unit
DTE	Data Terminal Equipment (computers, terminals or printers)
FCS	Frame Check Sequence
FDDI	Fibre Distributed Data Interface
FIPS	Federal Information Processing Standard
FRMR	Frame Reject (a HDLC frame type)
FTP	File Transfer Protocol
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
GSM	Global System for Mobile communications
HCS	Header Check Sequence
HDLC	High-level Data Link Control
HHU	Hand Held Unit
HLS	High Level Security
HMAC	Keyed-Hash Message Authentication Code
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
.ind	.indication service primitive
IP	Internet Protocol
ISO	International Organization for Standardization

IV	Initialization Vector
LAN	Local Area Network
LDN	Logical Device Name
LLC	Logical Link Control (Sublayer)
LLS	Low Level Security
L-SAP	LLC sublayer Service Access Point
LPDU	LLC Protocol Data Unit
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)
master	Central station – station which takes the initiative and controls the data flow
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
NDM	Normal Disconnected Mode
NIST	National Institute of Standards and Technology
NRM	Normal Response Mode
N(R)	Receive sequence Number
N(S)	Send sequence Number
o	optional, used in conjunction with attribute and method definitions
OBIS	Object Identification System
OSI	Open System Interconnection
PAR	Positive Acknowledgement with Retransmission
PDU	Protocol data unit
P/F	Poll/Final
PhL	Physical Layer
PHPDU	PH PDU
PHSDU	PH SDU
PLC	Power line carrier
PSDU	Physical layer Service Data Unit
PSTN	Public Switched Telephone Network
PPP	Point-to-Point Protocol
RARP	Reverse Address Resolution Protocol
.req	.request service primitive
.res	.response service primitive
RLRE	A-Release Response – an APDU of the ACSE
RLRQ	A-Release Request – an APDU of the ACSE
RNR	Receive Not Ready (a HDLC frame type)
RR	Receive Ready (a HDLC frame type)
SAP	Service Access Point

SDU	Service Data Unit
SNMP	Simple Network Management Protocol
SNRM	Set Normal Response Mode (a HDLC frame type)
server	A station, delivering services. The tariff device (meter) is normally the server, delivering the requested values or executing the requested tasks.
slave	Station responding to requests of a master station. The tariff device (meter) is normally a slave station.
TCP	Transmission Control Protocol
TDEA	Triple Data Encryption Algorithm
TL	Transport Layer
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
VAA	Virtual Application Association
V(R)	Receive state Variable
V(S)	Send state Variable
WPDU	Wrapper Protocol Data Unit
xDLMS_ASE	Extended DLMS Application Service Element

4. The DLMS/COSEM communications framework

4.1 Client/server type operation, communication profiles

Data exchange between data collection systems and metering equipment using the COSEM interface object model is based on the **client/server** paradigm. Metering equipment [1] play the role of the server.

To allow servers notifying clients about events, unsolicited services are also available.

Concepts, names and names terminology used below relate to the Open System Interconnection Reference Model described in ISO/IEC 7498-1. Their use is outlined in this clause and further developed in other clauses.

The data collection application and the metering application are modelled as one or more application processes, APs; see 4.5. Therefore, in this environment communication takes place always between a client and a server AP: the client AP requests services and the server AP provides them.

A client AP may be able to exchange data with a single or with multiple server APs at the same time. A server AP may be able to exchange data with one or more client APs at the same time.

NOTE Data exchange between server APs – the logical devices within a physical device – may be possible. Similarly, data exchange between client APs hosted by a single or multiple physical devices may be possible. Such exchanges are out of the scope of this specification.

Data exchange takes place via exchanging messages (SERVICE.requests / .responses) between the peer APs. In general, the client and the server APs are located in separate devices; exchanging messages therefore is done via a protocol stack, or communication profile, as shown in Figure 2.

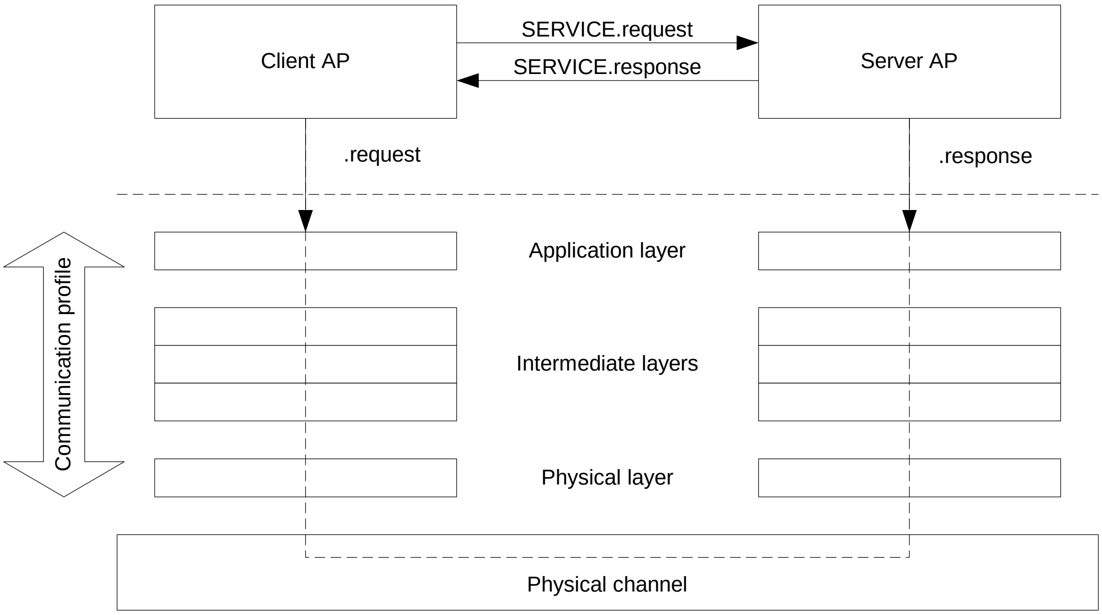


Figure 2 – Client/server relationship and protocols

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 AL. This is the only protocol layer containing COSEM specific element(s): the xDLMS_ASE, providing COSEM interface object related services. It may in principle be supported by any layer capable of providing the

[1] The term "metering equipment" is an abstraction; consequently the equipment playing the role of a server may be any type of equipment for which this abstraction is suitable.

services required by the COSEM AL. The number and type of lower layers depend on the communication media used.

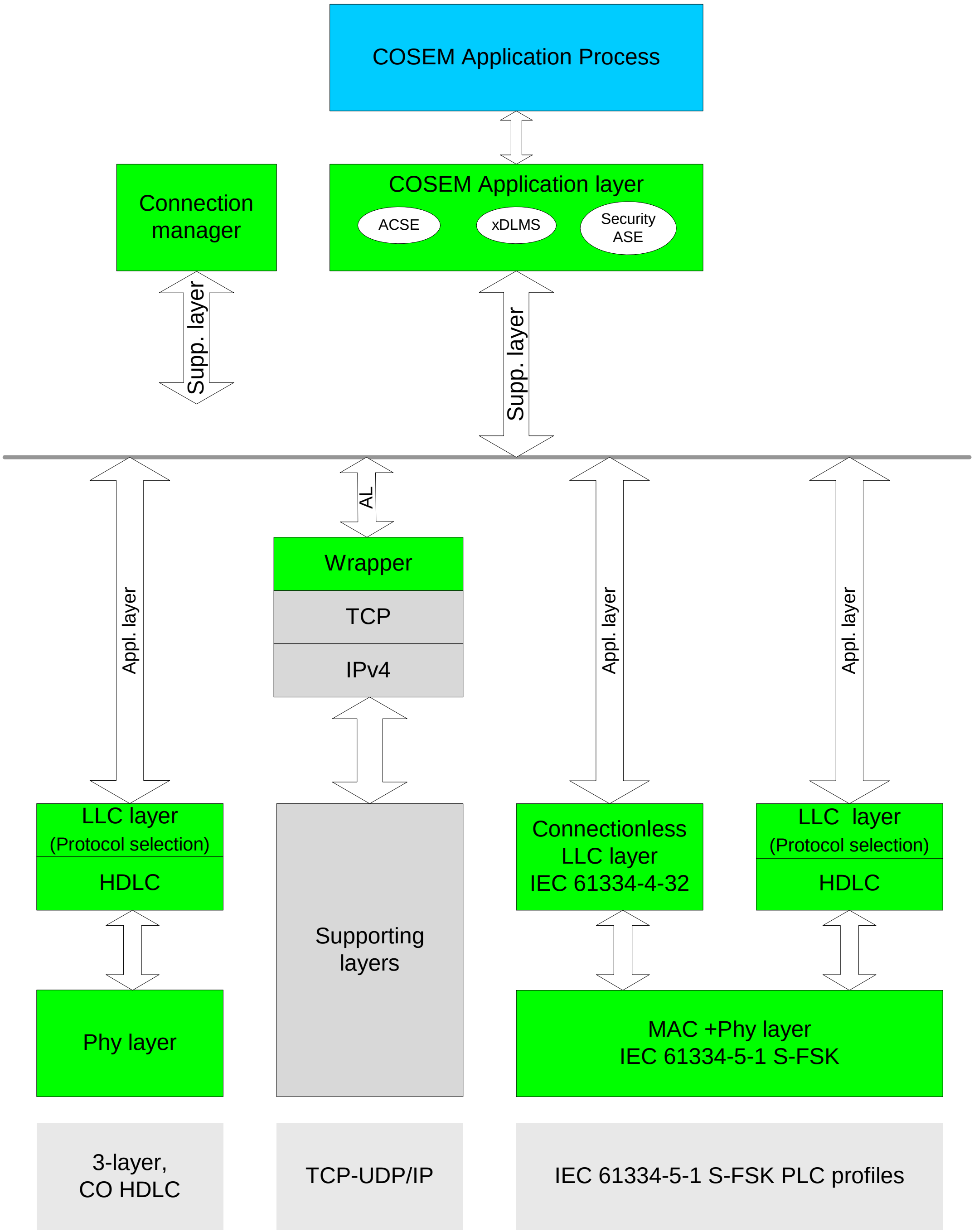


Figure 3 – DLMS/COSEM communication profiles

A given set of protocol layers with the COSEM AL on top constitutes a DLMS/COSEM communication profile. Each profile is characterized by the protocol layers included, their parameters, and by the type of the ACSE – connection-oriented or connectionless – included in the AL. A single device may support more than one communication profiles, to allow data exchange using various communication media. It is the task of the client side AP to decide which communication profile should be used.

This edition of the Green Book specifies the following communication profiles; see Figure 3.

- the 3-layer, connection-oriented (CO), HDLC-based profile. This comprises the COSEM AL, the HDLC-based data link layer and the PhL, for connection-oriented asynchronous data exchange. It supports data exchange via a local – optical or electrical – port according to IEC 62056-21, leased lines and the PSTN or the GSM telephone network. See 8 and 10.2;

- the TCP-UDP/IP based communication profiles. These profiles support data exchange via the Internet over various physical media, like Ethernet, ISDN, GPRS, PSTN or GSM using PPP etc. In these profiles, the COSEM AL is supported by the COSEM transport layer(s), comprising a wrapper and the Internet TCP or UDP protocol. Lower layers can be selected according to the media to be used, as the TCP-UDP layers hide their particularities. See 7 and 10.3;
- the S-FSK PLC based communication profiles. These profiles support data exchange via power lines using S-FSK modulation. In these profiles, the COSEM AL is supported by the connectionless LLC sublayer as specified in IEC 61334-4-32, or the LLC sublayer using the data link layer based on the HDLC protocol as specified in IEC 62056-46 / clause 8 The MAC and the Physical layers are as specified in IEC 61334-5-1, with the extensions specified in this document.

Further communication profiles to support other media can be easily developed. The elements to be specified in each profile are given in 10.1.

4.2 Connection (association) oriented operation

The xDLMS application protocol is a connection-oriented (CO) protocol. It means, that the client and server APs can use the services of the xDLMS ASE only when they are “associated” [2]. In this environment a communication session consists of three phases, as it is shown in Figure 4:

- to be able to exchange application data, an application level connection, called Application Association, has to be established between a client and a server AP. This is the task of the connection-oriented ACSE of the AL. 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 necessary AA is established, application data exchange can take place using the services provided by the xDLMS service element;
- at the end of the data exchange, the AA has to be released.

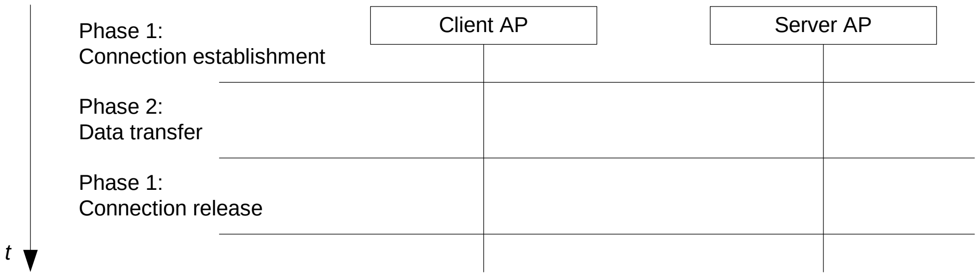


Figure 4 – A complete communication session in the CO environment

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 data transfer phase: it can be considered that the connection establishment phase has been already done somewhere in the past. Pre-established AAs cannot be released.

4.3 Interoperability and interconnectivity in COSEM

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

[2] Application associations can be considered as application level connections.

Interoperability is an application level notion: a client AP is interoperable with a server AP, if it is able to establish AAs using the A-Associate service of the standard connection-oriented ACSE, as it is specified in Clause 9.

AAs may be established between a client AP and server APs using various application contexts, authentication mechanisms and xDLMS contexts as well as other parameters. For example, a client AP may establish an AA with a server AP with an application context using short name (SN) referencing and with another server AP with an application context using logical name (LN) referencing. Although the messages exchanged depend on the application context of the AA established, both server APs are interoperable with the client AP if it is able to establish the AA using the right context with both server APs. With this, using the services of the standard ACSE for AA establishment ensures interoperability in COSEM. In the case of pre-established AAs – see 4.2 – the choice of all parameters has in effect been previously made and fixed, for each AA.

On the other hand, *interconnectivity* is a protocol level notion: in order to be able to exchange messages, the client and the server APs should be ***interconnectable*** and ***interconnected***.

Before the two APs can establish an AA, they must be *interconnected*. The two APs 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 APs should be interconnectable and shall establish the required connections. Two APs are *interconnectable* if they use the same communication profile.

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

4.4 Ensuring interconnectivity: the protocol identification service

In DLMS/COSEM, AA establishment is always initiated by the client AP. 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 AP must obtain information about the protocol stack implemented in the server.

The COSEM framework provides a specific, application level service to for this purpose: the protocol identification service. It is an optional application level service, allowing the client AP to obtain information – after establishing a physical connection – about the protocol stack implemented in the server. The protocol identification service, 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 acces to the PhL.

4.5 Application models

COSEM models metering equipment as a set of logical devices, hosted in a single physical device. Each logical device 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 interface objects.

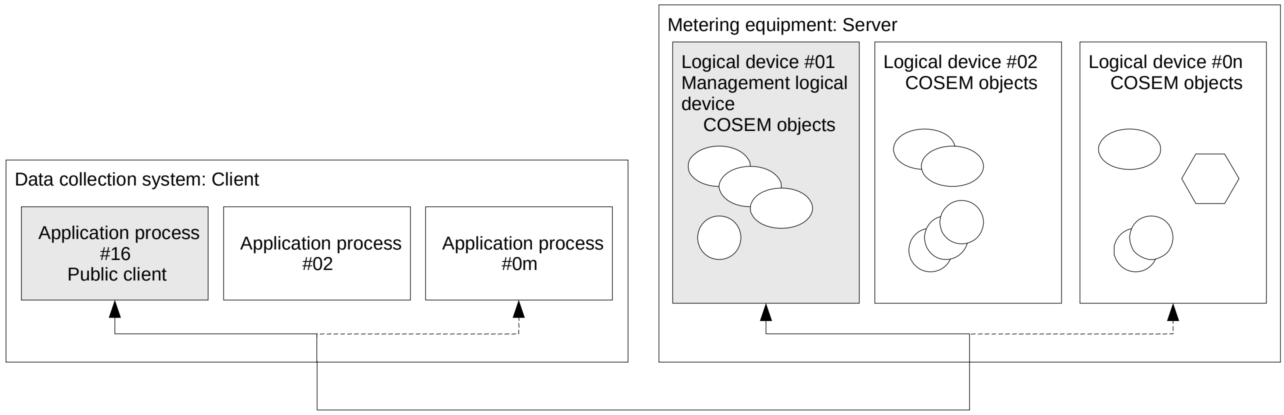

Metering equipment: Server
Logical device #01 Management logical device COSEM objects
Logical device #02 COSEM objects
Logical device #0n COSEM objects
Data collection system: Client
Application process #16 Public client
Application process #02
Application process #0m

Figure 5 – COSEM application model of a data collection system and metering equipment

DCSs are modelled as a set of APs. Each AP may have different roles and access rights, granted by the metering equipment.

NOTE The application processes may be hosted by one or several physical devices.

The Public client and the Management logical device APs have a special role and they must always be present. See more in the clause “COSEM interface classes” in the Blue Book, DLMS UA 1000-1.

4.6 Model of DLMS/COSEM servers

Figure 6 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.

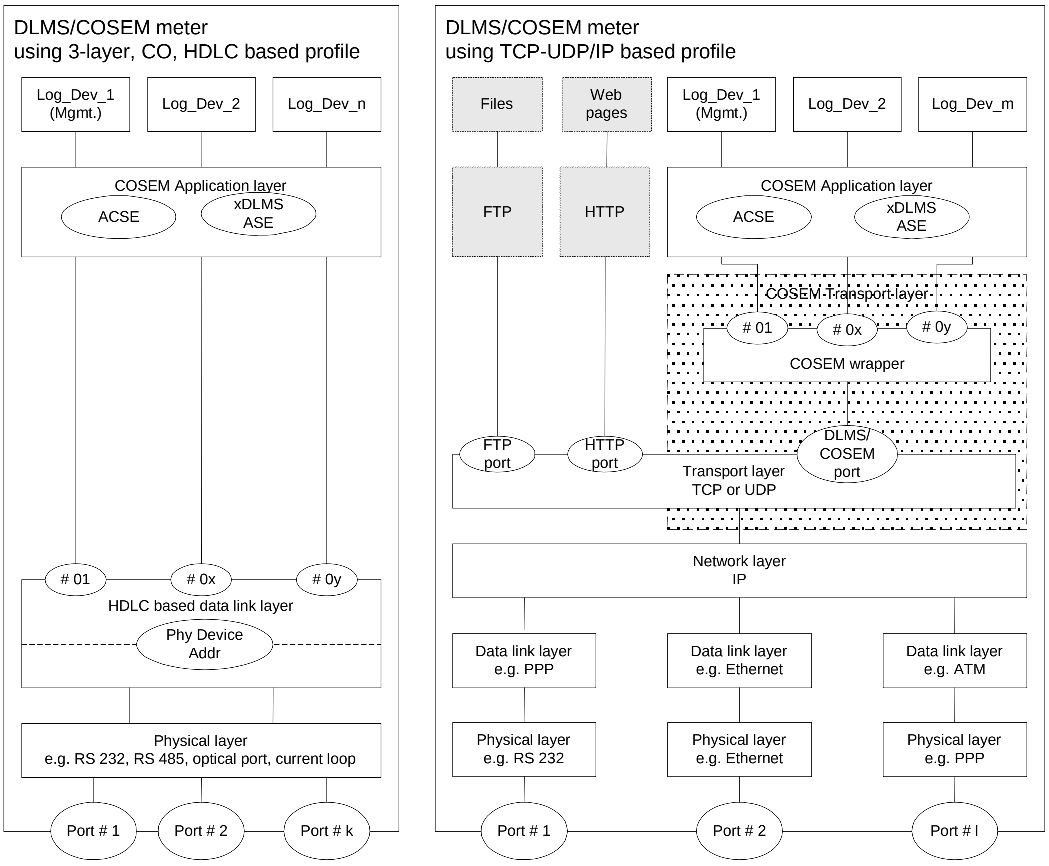


Figure 6 – DLMS/COSEM server model

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

The 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 LAN 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.

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 COSEM AL is supported by the 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 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.

Through the wrapper, the 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 etc.).

Obviously, in a single server it is possible to implement several protocol stacks, with the common 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.

4.7 Model of a DLMS/COSEM based client

Figure 7 shows the model of a DLMS/COSEM based client as an example.

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

- in this particular model, the COSEM AL is supported either by the HDLC-based data link layer or the 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 we have seen, client APs and server logical devices 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 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.

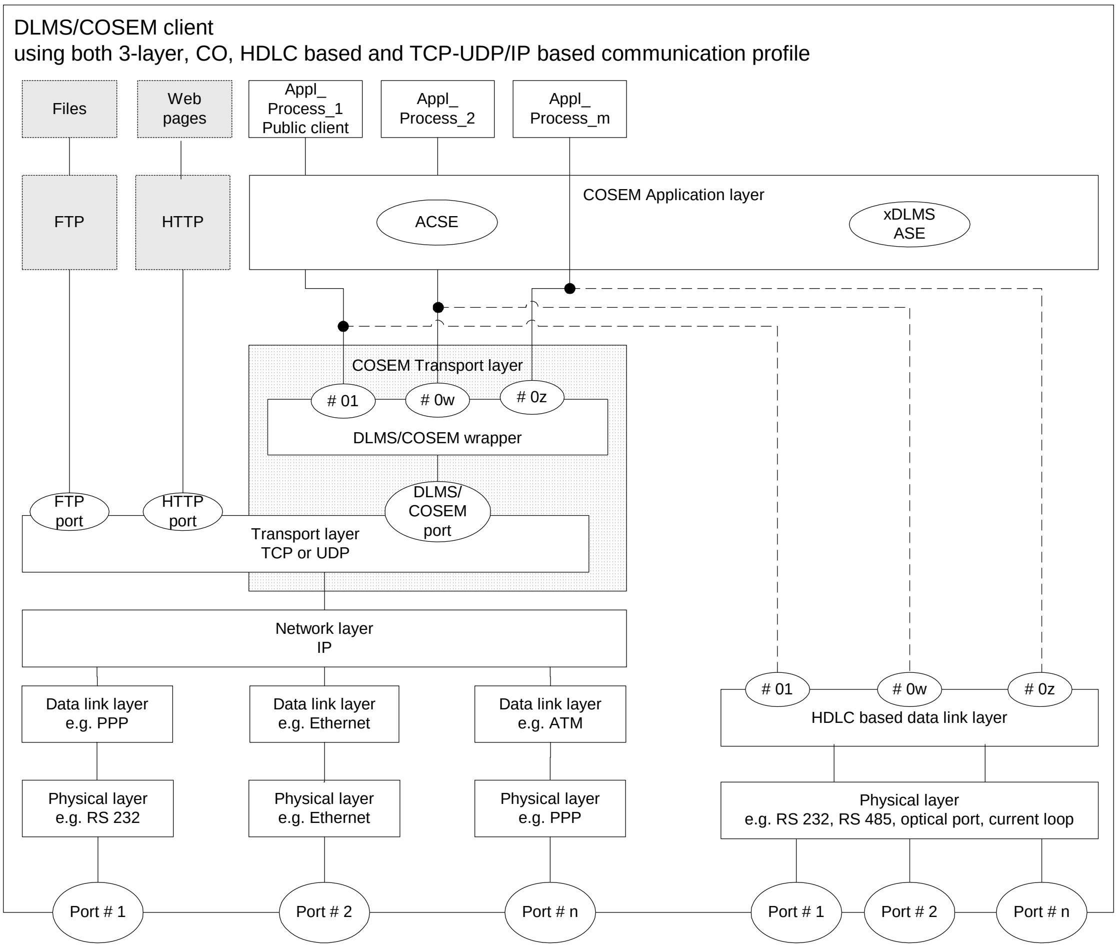


Figure 7 – Model of a DLMS/COSEM based client using multiple protocol stacks

4.8 Model of a DLMS/COSEM based data collection system

Figure 8 shows the model of a DLMS/COSEM based data collection system, consisting of two metering sites and two remote data collection systems is shown.

On metering site #1, metering equipment use the TCP-UDP/IP based communication profile and they are connected to an Ethernet LAN. In addition, a local data collection system (DCS) is installed. Each physical device has its own IP address. The entry point to the metering site is the same as that of the LAN. Meters can be reached remotely through the internet, locally through the local DCS, or directly by a Hand Held Unit (HHU). On the optical port, meters may communicate either using the 3-layer, CO, HDLC-based communication profile or the TCP-UDP/IP based communication profile using PPP.

On metering site #2, metering equipment use the 3-layer, CO, HDLC-based communication profile. To be able to reach them through a single WAN access point, they are connected to a bus, for example RS 485. The address of a physical device on the LAN is provided by its lower HDLC address. As RS 485 does not provide a protocol for handling collisions on the bus, the client may exchange data with the servers on the LAN one by one. In other words, the task of the bus master is performed by the client. The access point of the LAN is a modem with an RS 485 interface. Its address is provided by the WAN, which can be either the PSTN or the GSM telephone network. For local data exchange, a portable DCS, connected directly to the RS 485 bus may be used. In this case, remote access may not be available during the local communication. Like at metering site #1, direct local data exchange is available using an HHU.

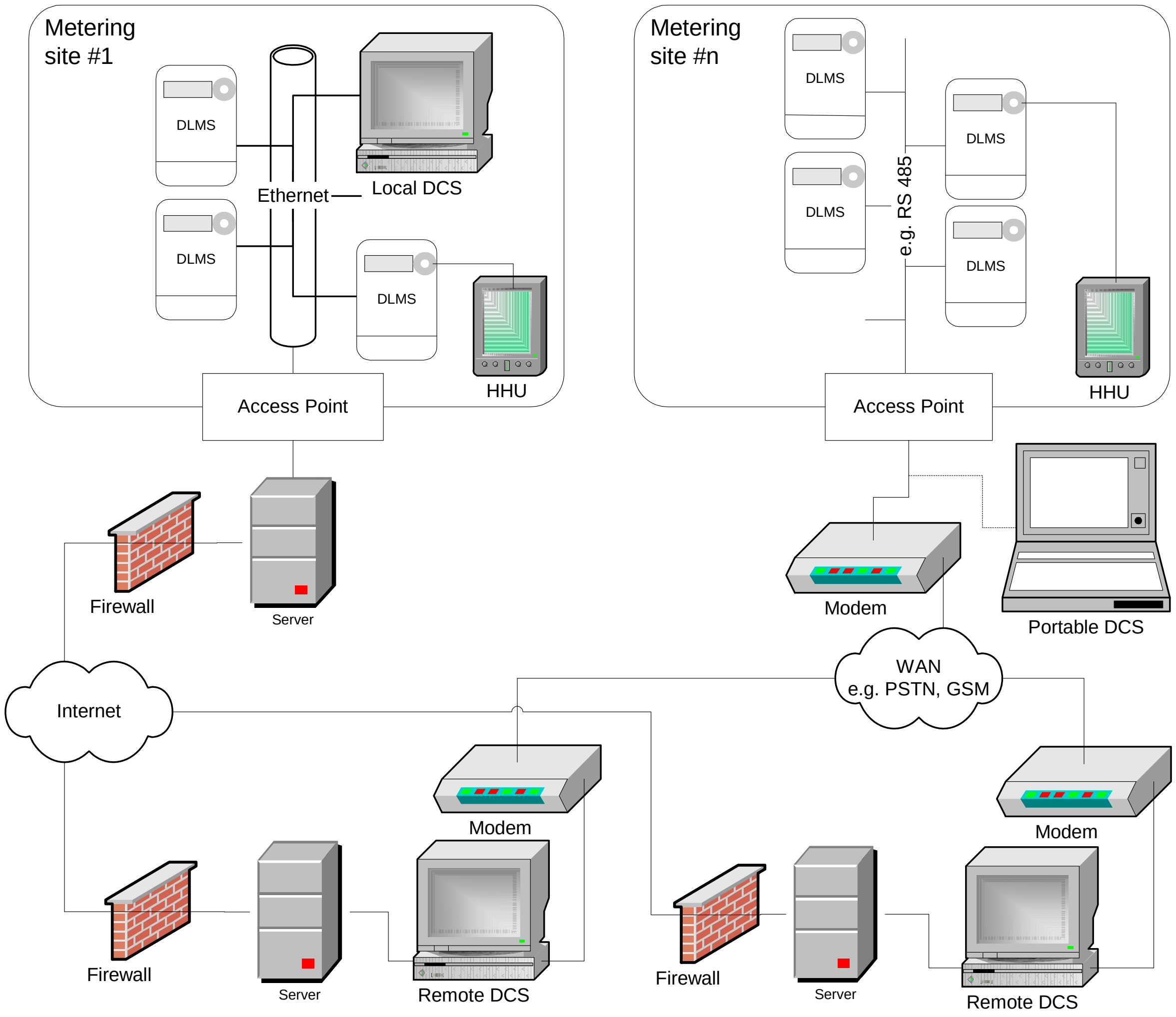


Figure 8 – Model of a DLMS/COSEM based meter data collection system

Other LAN types, like M-Bus (EN 13757-2), Euridis (IEC 62056-31) or PLC are also possible.

The architecture of the two remote DCSs is identical. Both systems can reach both metering sites via either the internet or a PSTN or GSM WAN.

The address of the physical device hosting the client APs is provided by the WAN. The AP address identifies the type of the client only; for example address 0x10 is the address of the Public Client in each DCS.

4.9 Access requirements

DLMS/COSEM meets the following access requirements for data exchange:

- it allows various parties (data collection systems) to have access to metering data;
- it allows to exchange data with a single or multiple metering equipment at a metering site;
- in the case of multiple metering equipment at a metering site, a single access point can be made available;
- it is possible to exchange data with metering equipment either remotely or locally;
- depending on the resources of the metering equipment, local and remote data exchange may be performed without interfering with each other;
- it is possible to use various communication media both on local area networks (LAN) and wide area networks (WAN);
- it provides authentication mechanisms to control access to data; these mechanisms are made available by the COSEM AL and the interface objects (Association object);

- it provides cryptographic protection of the messages;
- it supports easy system integration and meter deployment;
- it provides a migration path from certain legacy systems to DLMS/COSEM based systems.

The key element to ensure that the above requirements are met is the AA provided by the COSEM AL. For details, see the relevant clauses of this book. Below, two aspects are dealt with in some detail.

4.10 System integration and meter installation

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

As shown in Figure 5, 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.

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 read the internal structure of the meter, and then download 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.

4.11 Migration

By today, there are thousands of data collection systems – based on legacy protocols, for example IEC 61107 – installed. Obviously, a migration path is necessary.

DLMS/COSEM provides this by adding a new protocol mode “E” to the IEC 62056-21 standard (was IEC 61107). With this, during the opening sequence, the meter (server) is able to advise the HHU (client) that the advanced Mode E is available. If the HHU acknowledges it, they will continue the data exchange using the 3-layer, CO, HDLC-based protocol. The information exchange takes place then using the COSEM object model. If not, data exchange continues in the conventional Mode C, although the functionality may be limited.

5. Physical layer services and procedures for connection-oriented asynchronous data exchange

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 companion specification.

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 10. Figure 9 shows a typical configuration for data exchange through a wide area network, for example the PSTN.

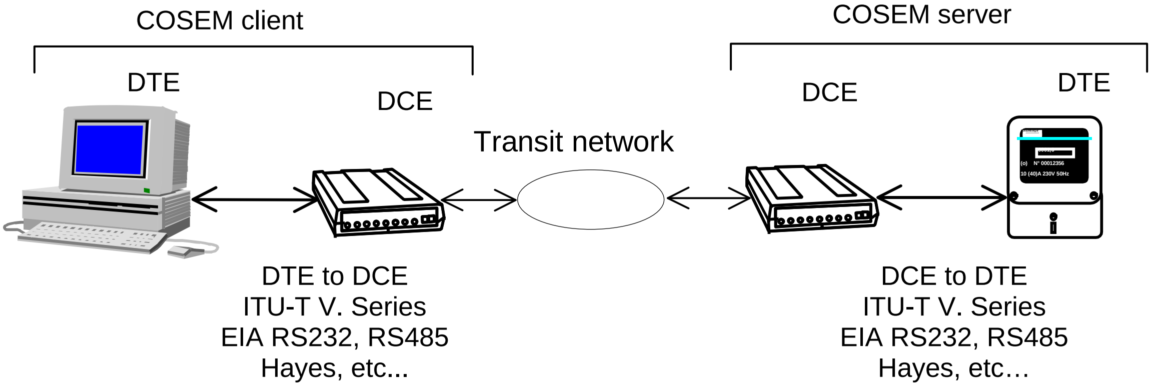


Figure 9 – Typical PSTN configuration

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 companion specification 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.

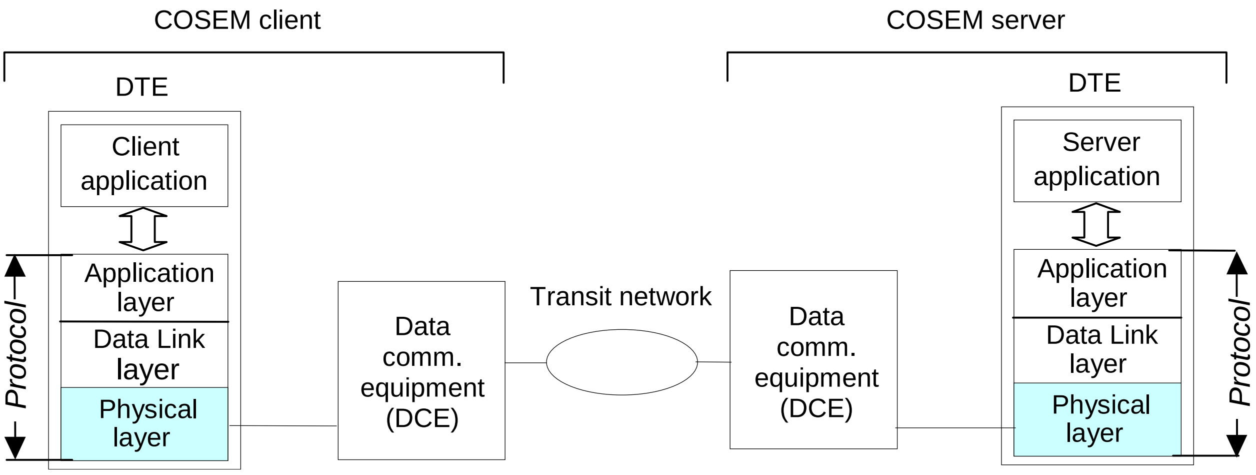


Figure 10 – The location of the physical layer

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 companion specification.

5.2.2 Use of the physical layer services

Figure 11 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.

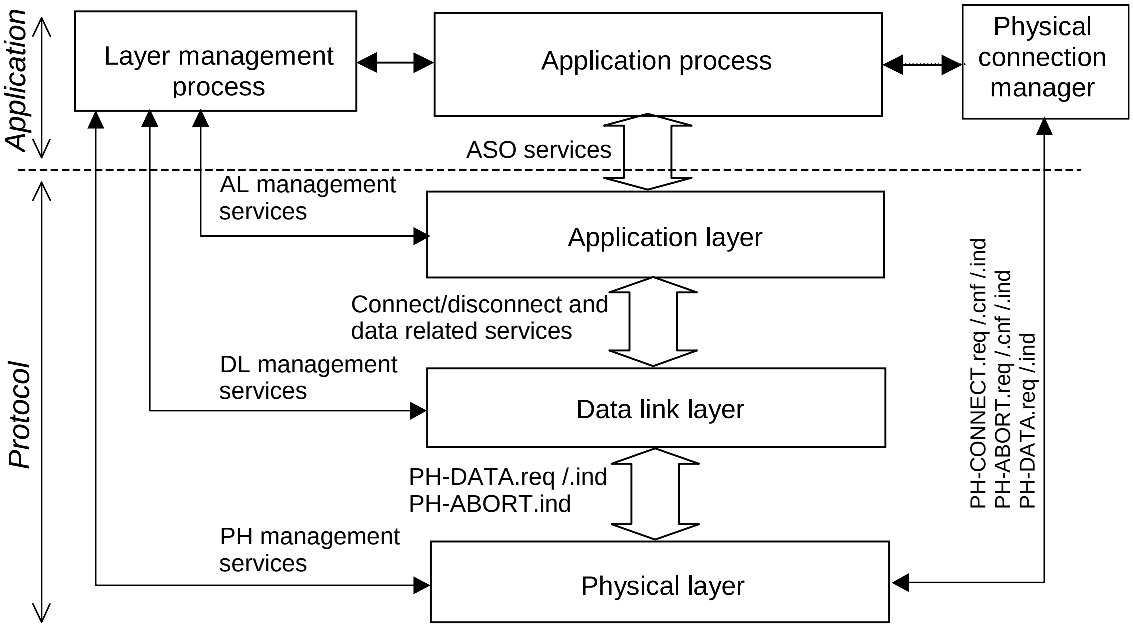


Figure 11 – Protocol layer services of the COSEM 3-layer connection-oriented profile

more details, see complete Green Book ....

6. Direct Local Connection (excerpt)

6.1 Introduction

This chapter 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 companion specification shall be used.

The switch to the baudrate 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.

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, Clause 6.3.14) [3].

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

6.3 Overview

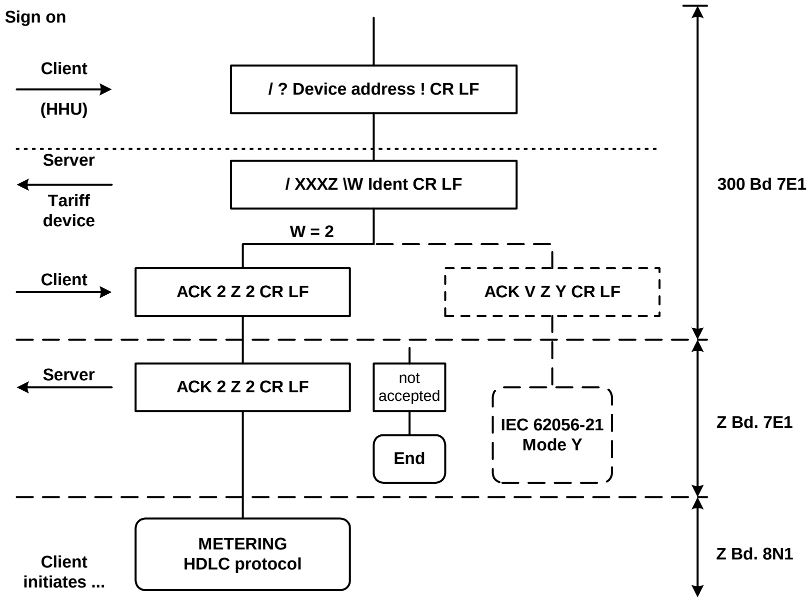


Figure 12 – Entering protocol mode E (HDLC)

more details, see complete Green Book ....

7. COSEM transport layers for IPv4 networks

7.1 Scope

This chapter specifies the transport layers (TLs) for COSEM communication profiles for use on IPv4 networks.

These COSEM_on_IP communication profiles contain a connection-less and a connection-oriented TL, providing OSI-style services to the service user COSEM AL. The connection-less TL is based on the User Datagram Protocol Internet standard STD0006. The connection-oriented TL is based on the Internet standard Transmission Control Protocol Internet standard STD0007.

Although the major part of the COSEM TLs is the UDP and TCP as they are specified in the relevant Internet standards, they include an additional sublayer, called wrapper.

Clause 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 COSEM_on_IP profiles, the COSEM AL uses the services of one of these TLs, which use then the services of the Internet Protocol (IPv4) network layer to communicate with other nodes connected to the abstract IPv4 network.

When used in these profiles, the 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 13.

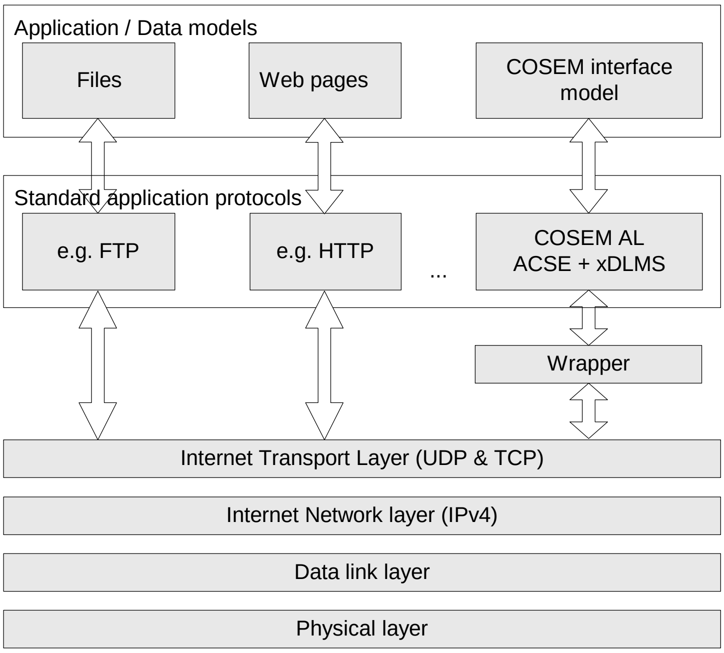

Application / Data models
Files
Web pages
COSEM interface model
Standard application protocols
e.g. FTP
e.g. HTTP
...
COSEM AL ACSE + xDLMS
Wrapper
Internet Transport Layer (UDP & TCP)
Internet Network layer (IPv4)
Data link layer
Physical layer

Figure 13 – 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 (last updated 2009-12-21)

• dlms/cosem	4059/TCP	DLMS/COSEM
• dlms/cosem	4059/UDP	DLMS/COSEM

As the COSEM AL specified in Clause 9 uses and provides OSI-style services, a wrapper has been introduced between the UDP/TCP layers and the COSEM AL. Therefore, the 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 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 COSEM AL is listening only on one UDP or TCP port. On the other hand, as defined in DLMS UA 1000-1, Clause 4.1.5, a COSEM physical device may host several client APs or server logical devices. The additional addressing capability provided by the wrapper sublayer allows identifying these APs.

The structure of the COSEM TL and their place in COSEM_on_IP is shown in Figure 14.

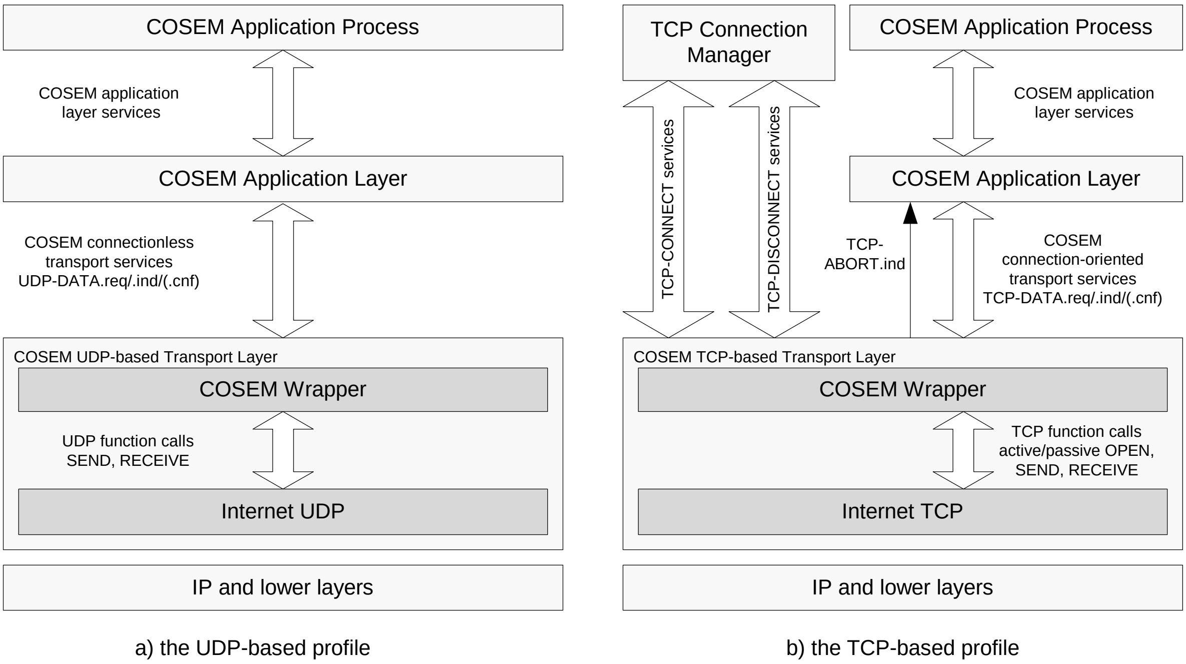

COSEM Application Process
COSEM application layer services
COSEM Application Layer
COSEM connectionless transport services UDP-DATA.req/.ind/(.cnf)
COSEM UDP-based Transport Layer
COSEM Wrapper
UDP function calls SEND, RECEIVE
Internet UDP
IP and lower layers
a) the UDP-based profile
TCP Connection Manager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

Figure 14 – Transport layers of the COSEM_on_IP profile

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

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

7.3.1 General

The 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, 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 of 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 COSEM UDP-based TL, then invoking confirmed COSEM 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 IPv4 address [4].

more details, see complete Green Book ....

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

7.4.1 General

The COSEM connection-oriented TL is based on the connection-oriented Internet transport protocol, called Transmission Control Protocol or TCP. 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, 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 COSEM TCP-based TL provides OSI-style services to the service user(s) for all three phases:

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

The COSEM connection-oriented, TCP-based TL contains the same wrapper sublayer as the 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 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 Clause 7.5.

more details, see 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.

[4] The addressing/identification scheme for the COSEM_on_IP profiles is defined in the complete Green Book.

According to the TCP connection state diagram (Figure 15) 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 16.

This process, consisting of three messages, establishes the TCP connection and “synchronizes” the initial sequence numbers [5] 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.

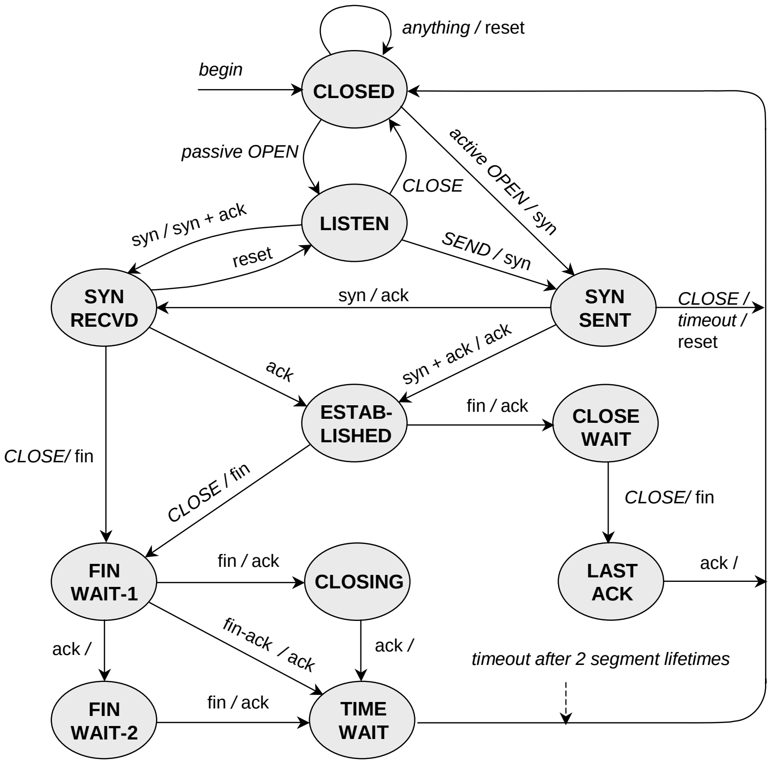
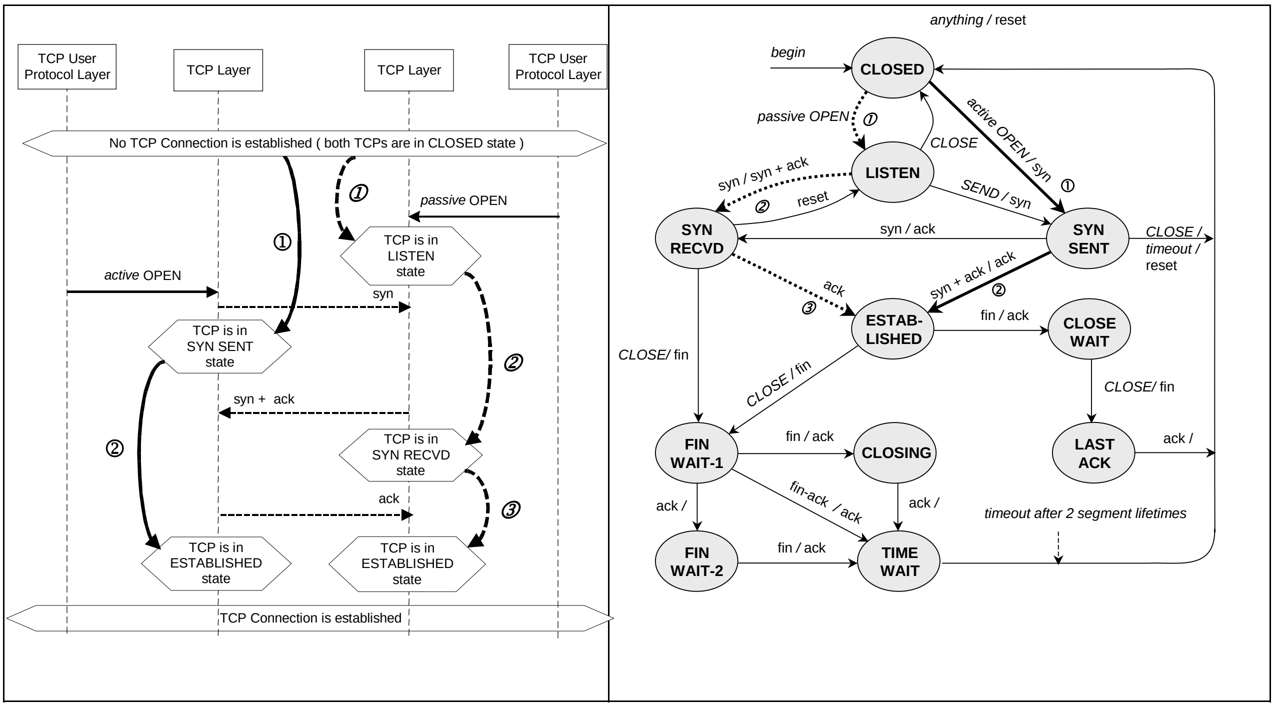


Figure 15 – TCP connection state diagram

[5] 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.



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

Figure 16 – MSC and state transitions for establishing a transport layer and TCP connection

more details, see 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 electricity 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 companion specification, the following selections from the HDLC standard have been made:

- unbalanced connection-mode data link operation [6];
- 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.

[6] 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 paradigm.

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 companion specification 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 sub-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.

more details, see 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 17.

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

Figure 17 – The ISO/IEC 8802-2 LLC PDU format

For the purposes of DLMS/COSEM, this LLC PDU format is used as shown on Figure 18:

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

Figure 18 – 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 last bit is used as a command/response identifier: z = 0 means ‘command’ and z = 1 means “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 User Association. 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.

more details, see 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 19:

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

Figure 19 – 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 sub-clauses.

8.4.1.2 Flag field

The length of the flag field is one byte and its value is $7E_H$. 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 20.

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 20 – 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 21:

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 21 – 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). See also the complete Green Book.

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 sub-clause 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 22. The server may support more than one addressing schemes. Individual, multicast and broadcast addressing facilities are provided for both the upper and the lower HDLC address.

	LSB
Upper HDLC address	1

	LSB		LSB
Upper HDLC address	0	Lower HDLC address	1
First byte		Second byte	

	LSB		LSB		LSB		LSB
Upper HDLC addr. high	0	Upper HDLC addr. low	0	Lower HDLC addr. high	0	Lower HDLC addr. low	1
First byte		Second byte		Third byte		Fourth byte	

Figure 22 – Valid server address structures

8.4.2.3 Reserved special HDLC addresses

The following special HDLC addresses are reserved:

Table 1 – Table of reserved client addresses

Reserved HDLC addresses	
0x00	No-station
0x01	Client Management Process
0x10	Public Client

Table 2 – Table of reserved server addresses

Reserved upper HDLC addresses		
One byte address	Two byte address	
0x00	0x0000	No-station
0x01	0x0001	Management Logical Device
0x02..0x0F	0x0002..0x000F	Reserved for future use
0x7F	0x3FFF	All-station (Broadcast)
Reserved lower HDLC addresses		
0x00	0x0000	No-station
0x01...0x0F	0x0001..0x000F	Reserved for future use
0x7E	0x3FFE	CALLING [7] Physical Device
0x7F	0x3FFF	All-station (Broadcast)

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 = $3A_H = 00111010_B$

Server HDLC Address (using four bytes addressing)

lower HDLC Address = $3FFF_H = 0011111111111111_B$ All-station (Broadcast) Address

upper HDLC Address = $1234_H = 0001001000110100_B$

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

Server address	Client address

[7] The meaning of the CALLING Physical Device is discussed in 0

Upper HDLC high		Upper HDLC low		Lower HDLC high		Lower HDLC low		HDLC address	
	LSB		LSB		LSB		LSB		LSB
0100100	**0**	**0110100**	**0**	**1111111**	**0**	**1111111**	**1**	**0111010**	**1**
First byte		Second byte		Third byte		Fourth byte		Fifth byte	
Destination address								Source address	

Figure 23 – 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 specification;
- 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.

more details, see complete Green Book ....

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 3.

Table 3 – 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.

Length of the DA field received	Length of the DA field expected	Behaviour
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.

more details, see complete Green Book ....

8.5 FCS calculation

8.5.1 Test sequence for the FCS calculation [8]

The example presented here shows the proper FCS value for a two-byte frame consisting of 0x03 and 0x3F. The complete resulting frame is $7E_H$ 03_H $3F_H$ $5B_H$ EC_H $7E_H$.

\/ – 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 the internet Request for Comments 1662 [9] that describes the PPP.

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.

more details, see complete Green Book ....

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

[9] RFC 1662, *PPP in HDLC-like Framing*, July 1994, W. Simpson.

9. COSEM application layer

9.1 Overview

9.1.1 COSEM application layer structure

The structure of the client and server COSEM application layers is shown in Figure 24.

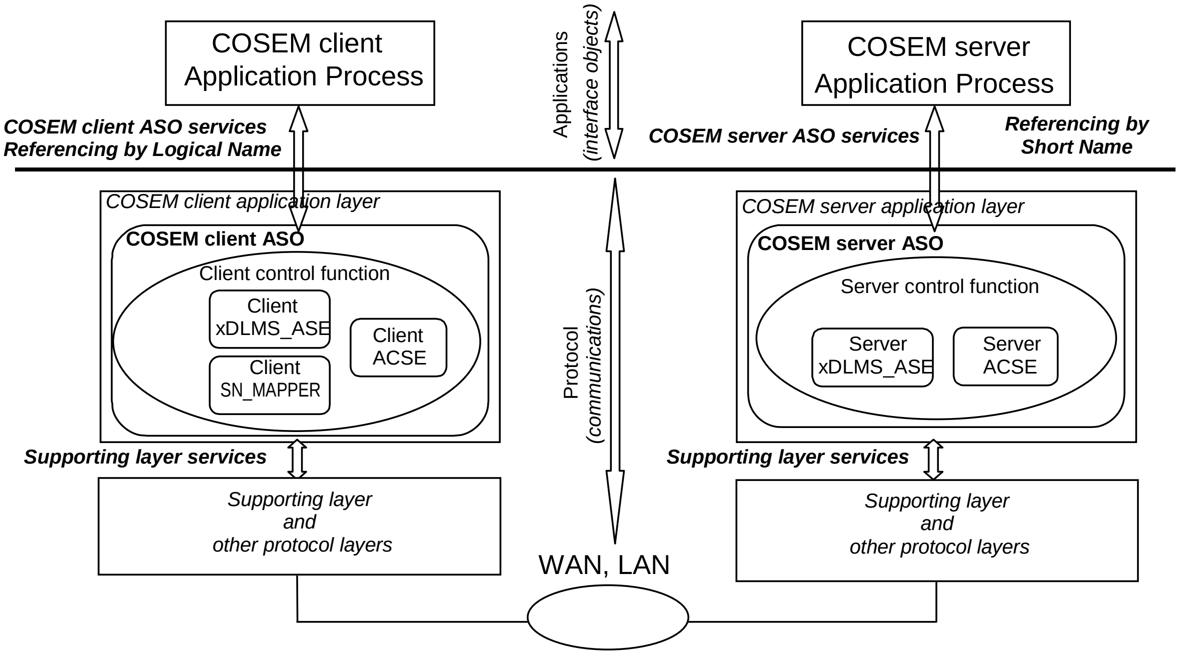

COSEM client Application Process
COSEM server Application Process
Applications (interface objects)
COSEM client ASO services Referencing by Logical Name
COSEM server ASO services
Referencing by Short Name
COSEM client application layer
COSEM client ASO
Client control function
Client xDLMS_ASE
Client ACSE
Client SN_MAPPER
Protocol (communications)
COSEM server application layer
COSEM server ASO
Server control function
Server xDLMS_ASE
Server ACSE
Supporting layer services
Supporting layer services
Supporting layer and other protocol layers
Supporting layer and other protocol layers
WAN, LAN

Figure 24 – The structure of the COSEM Application layers

The main component of the COSEM AL is the COSEM Application Service Object. It provides services to its service user, the COSEM Application Process, and uses services provided by the supporting lower 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 SN_MAPPER ASE.

The task of ACSE is to establish, maintain, and release application associations. For the purposes of DLMS/COSEM connection oriented (CO) communication profiles, the CO ACSE, specified in ISO/IEC 8649 and ISO/IEC 8650-1 is used.

The task of the xDLMS_ASE is to provide data transfer services between COSEM APs. It is based on the DLMS standard, IEC 61334-4-41. It has been extended for DLMS/COSEM; see 9.1.2.3.1. The main objective of DLMS/COSEM is to provide a business domain oriented interface object model for metering devices and systems while keeping backward compatibility to the DLMS standard. To meet these objectives, DLMS/COSEM includes an evolution of DLMS. Remaining fully compliant to the DLMS standard, DLMS/COSEM provides a more metering specific view of the meter through the COSEM interface objects.

DLMS UA 1000-1 Clause 4 specifies two referencing methods to attributes and methods of COSEM interface objects for use in COSEM servers: LN and SN referencing. Therefore, on the server side, two distinct xDLMS service sets are specified: one exclusively for LN referencing and the other exclusively for SN referencing. It can be considered that there are two different xDLMS_ASEs: one providing services for LN referencing and the other for SN referencing. The server AL may include one, the other or both xDLMS_ASEs.

The 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.

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

The optional 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 also the complete Green Book.

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

- encoding the ACSE APDUs in BER (ISO/IEC 8825);
- encoding the xDLMS APDUs carrying the data transfer services in A-XDR (IEC 61334-6).

9.1.2 COSEM application layer services

9.1.2.1 ASO services

The services required by, or provided for the COSEM client and server APs at the logical interfaces with the respective COSEM AL, using CO procedures, fall into three categories:

- application association establishment and release;
- data transfer;
- layer management.

9.1.2.2 Services provided for application association establishment and release

The services provided for the establishment and release of AAs 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 used 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. 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. Therefore, 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 the complete Green Book, the COSEM-RELEASE service in the complete Green Book and the COSEM-ABORT service in the complete Green Book..

9.1.2.3 Services provided for data transfer

9.1.2.3.1 The xDLMS application service element

As mentioned in 9.1.1, xDLMS includes some extensions to the DLMS standard, IEC 61334-4-41. These extensions define added functionality; existing functionality is not modified. They are made in such a way, that there is no conflict with the existing DLMS standard and comprise the following:

- additional services;
- additional data types;
- new DLMS version number;
- new conformance block;
- clarification of the meaning of the PDU size.

The additional services are GET, SET, ACTION and EventNotification. They allow to reference attributes and methods of COSEM interface objects using LN referencing. See 9.1.2.3.5 and 9.1.2.3.6.

The additional data types are specified in the complete Green Book.

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

The new 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 the complete Green Book.

For services using SN referencing, new variants of the Variable_Access_Specification service parameter, the Read.response and the Write.response services have been added to support selective access and block transfer.

To clarify the meaning of the maximum PDU size usable by the client and the server, the modifications shown in Table 4 have been made. The xDLMS-Initiate service uses these names for PDU sizes.

Table 4 – Clarification of the meaning of PDU size for DLMS/COSEM

was:	**new:**
Page 61, Table 3 of IEC 61334-4-41:	
Proposed Max PDU Size	Client Max Receive PDU Size
Negotiated Max PDU Size	Server Max Receive PDU Size
Page 63, 5th paragraph of IEC 61334-4-41:	
The Proposed Max PDU Size parameter, of type Unsigned16, proposes a maximum length expressed in bytes for the exchanged DLMS PDUs. 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 PDU 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 the there is no limit on the PDU size.
Page 63, last paragraph of IEC 61334-4-41:	
The Negotiated Max PDU Size parameter, of type Unsigned16, contains a maximum length expressed in bytes for the exchanged DLMS PDUs. 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 PDU 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 the there is no limit on the PDU size.

9.1.2.3.2 Client/server and non-client/server type services

xDLMS data transfer services are related to attributes and methods of COSEM interface objects, specified in DLMS UA 1000-1, and provide the interface between the COSEM application model and the communication protocols.

When the server uses SN referencing, attributes and methods of COSEM interface objects are mapped to DLMS named variables. The mapping is held by the Association SN interface object(s).

For accessing attributes and methods, client/server type services are used: the client requests services and the server provides them.

There is also an unsolicited, non-client/server type service. This service is requested by the server, upon an occurrence of an event, to inform the client of the value of one or more attributes, as though they had been requested by the client. It is an unconfirmed service.

9.1.2.3.3 Referencing methods and service mapping

As specified in 9.1.1, there are two distinct service sets available, one for LN and one for SN referencing.

On the client side, in order to handle the different referencing methods transparently for the AP, the AL provides only one service set, using LN referencing. Using a unique, standardized service set between COSEM client APs and the communication protocol – hiding the particularities of COSEM servers using different referencing methods – allows to specify 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 services using LN referencing or services using SN referencing or both kind of services can be provided.

In case of confirmed AAs, the referencing method and the service set to be used is negotiated during the AA establishment phase via the COSEM application context, see 9.4.2.2.2, and the conformance block, see the complete Green Book. It shall not change during the lifetime of the AA established. Using LN or SN services within a given AA is exclusive.

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

When the server uses LN referencing, the services are the same on both side. When the server does not use LN referencing, a mapping between client side and server side services takes place. As explained in 9.1.1, if the server uses SN referencing this mapping is performed by the Client SN_MAPPER ASE. See also the complete Green Book.

9.1.2.3.4 Confirmed and unconfirmed services

Within confirmed AAs, client/server type data transfer services can be invoked in a confirmed or unconfirmed manner. Within unconfirmed AA-s, client/server type data transfer services may be invoked in an unconfirmed manner only. See also the complete Green Book.

9.1.2.3.5 COSEM client/server type services

COSEM client/server type data transfer services for LN referencing are the following:

- the GET service, used to read the value of one or more attributes of COSEM interface objects, see the complete Green Book;
- the SET service, used to write the value of one or more attributes of COSEM interface objects, see the complete Green Book;
- the ACTION service, used to invoke one or more methods of COSEM interface objects. Invoking methods may imply sending service parameters and returning data, see the complete Green Book.

COSEM client/server type data transfer services for SN referencing are the following:

- the Read service, used to read the value of one or more attributes or to invoke one or more methods of COSEM interface objects when return parameters are expected. It is a confirmed service. See the complete Green Book;
- the Write service, used to write the value of one or more attributes or to invoke one or more methods of COSEM interface objects when no return parameters are expected. It is a confirmed service. See the complete Green Book.

- the UnconfirmedWrite service, used to write the value of one or more attributes or to invoke one or more methods of COSEM interface objects. It is an unconfirmed service. See the complete Green Book.

9.1.2.3.6 Services for event notification

To support event notification, COSEM also provides non client/server type, unsolicited services:

- with LN referencing the EventNotification service, see the complete Green Book.;
- with SN referencing, the InformationReport service, see the complete Green Book..

9.1.2.3.7 Identifying 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. 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.

The EventNotification service – being a non-client/server type service – does not contain the Invoke_Id parameter.

This feature is available only with LN referencing.

9.1.2.3.8 Priority of service invocations: the Priority parameter

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 is served before the previous requests with priority NORMAL. The response carries the same priority flag as that of the corresponding request. Managing priority is a negotiable feature; see the complete Green Book.

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 shall be served with NORMAL priority.

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

9.1.2.3.9 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. 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 these access selection parameters. In the case of LN referencing, this feature is called Selective access.

9.1.2.3.10 Multiple references

In a data transfer service invocation, it is possible to reference one or several attributes resp. methods. Using multiple references is a negotiable feature.

9.1.2.3.11 Attribute_0 referencing

With the GET and SET services, a special feature, Attribute_0 referencing is available. By convention, attributes of COSEM interface objects are numbered from 1 to n, where Attribute_1 is the logical name of the COSEM interface 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 and with the SET service in the complete Green Book.

NOTE As specified in DLMS UA 1000-1 sub-clause 4.1.1, manufacturers may add proprietary methods and/or attributes to any object, using negative numbers.

Attribute_0 referencing is a negotiable feature, see the complete Green Book.

9.1.2.3.12 Transferring long service parameters

The data transfer services are carried by the APDUs, exchanged between the peer ALs, in an encoded form. In some cases, the data may be longer then the Client / Server Max Receive PDU Size negotiated. To transfer such 'long' data, two transporting mechanisms are available:

a) long data transfer using the AL block transfer protocol. This mechanism can be used with any confirmed client/server type services. It is a negotiable feature, see the complete Green Book;

b) long data transfer in a transparent manner to the AL. This feature can be used when the supporting layer(s) provide(s) segmentation; see 10.

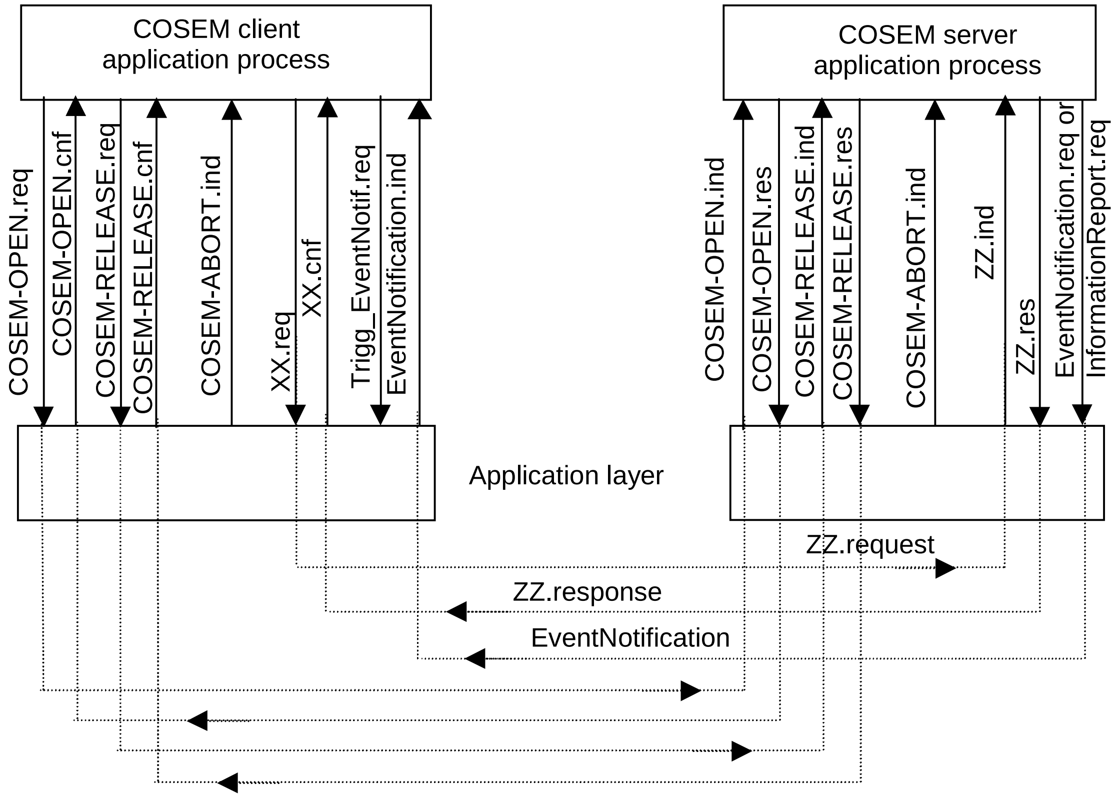
NOTE There is no mechanism within AL block transfer for re-sending or requesting a missing or corrupt block. Such a mechanism may or may not be a feature of long data transfer managed transparently by the supporting layers.

9.1.2.4 Layer management services

Layer management services have local importance only. Therefore, specification of these services is not within the scope of this companion specification. The specific SetMapperTable service is defined in the complete Green Book.

9.1.2.5 Summary of COSEM application layer services

A summary of the services available at the top of the COSEM AL is shown in Figure 25. Layer management services are not shown. Although the service primitives are different on the client and server side, the APDUs are the same. The abstract syntax of the ACSE and COSEM APDUs is specified in the complete Green Book.



NOTE XX and ZZ refer to client/server type data transfer services. These services may be different on the client side and the server side, if the server does not use LN referencing. See the complete Green Book.

Figure 25 – Summary of COSEM AL services

9.1.3 COSEM application layer protocols

The 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 AL protocol is based on the ACSE and the DLMS protocols, as specified in ISO/IEC 8650-1 and in IEC 61334-4-41 – with the extensions for DLMS/COSEM – respectively. 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 abstract syntax of the APDUs using ASN.1 as specified in ISO/IEC 8824.

The COSEM AL protocols are specified in 9.4.

9.2 Information security in DLMS/COSEM

Acknowledgement

This clause 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.

9.2.1 Definitions

9.2.1.1
confidentiality
preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information [FIPS PUB 199]

9.2.1.2
integrity
guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information [FIPS PUB 199]

9.2.2 Introduction

Confidentiality and integrity are among the key requirements when open systems connect to public media. With the increase of computational power, the requirements for strong cryptographic methods also increase. DLMS/COSEM provides two main information security features for accessing and transporting data:

- *data access security* controls access to the data held by a DLMS/COSEM server, see 9.2.3;
- *data transport security* allows the sending party to apply cryptographic protection to the xDLMS APDUs sent to ensure confidentiality and integrity. This requires ciphered ADPUs. The receiving party can remove or check this protection; see 9.2.4.

These information security features are provided partly by the COSEM AL, partly by COSEM objects:

- upon AA establishment, two contexts are negotiated using the ACSE services: the *application context* and the *authentication context*. The application context determines whether ciphered APDUs can be used or not. The authentication context determines the level of data access security. The ACSE APDUs are not cryptographically protected;

  NOTE The user-information field of the ACSE APDUs may be cryptographically protected.

- once an AA is successfully established, COSEM services can be used to access attributes and methods of COSEM objects, depending on the access rights valid in the given association. These services are carried by xDLMS APDUs, which may be cryptographically protected: authenticated, encrypted, or both, depending on the security policy in force. The cryptographic protection is applied, removed or checked by the COSEM AL. The COSEM AP of the sending party informs, via service parameters, the COSEM AL about the protection to be applied to the APDU to be sent. The COSEM AL of the receiving party informs the COSEM AP about the protection that has been applied to the APDU received.

The message security methods described in this document have been selected from the standards specified by NIST and the IETF.

For an overview of cryptography, see 15.

9.2.3 Data access security

9.2.3.1 Overview

Data access security concerns role based access to data in a DLMS/COSEM device.

It is managed by the Association LN / Association SN objects. Each COSEM server i.e. a logical device may support AAs with various clients, each having a different role, and with this, different access rights. Each AA is identified with a pair of lower layer addresses. Each Association object provides a list of objects visible in that particular AA and the access rights to their attributes and methods.

To be able to access data, the client must be properly authenticated. Upon AA establishment, an authentication context is negotiated between the client and the server. This specifies the required authentication of the peers, and, where needed, the security algorithm to verify the authentication. Three data access security levels are provided:

- Lowest level security (no security);
- Low Level Security (LLS);
- High Level Security (HLS).

9.2.3.2 Lowest level security (no security)

The purpose of lowest level security is to allow the client to retrieve some basic information. This authentication context does not require any peer authentication; it allows direct access to the data in the server, within the access rights available in the given AA.

9.2.3.3 Low Level Security (LLS)

The purpose of Low Level Security is to allow the authentication of clients by verifying the password supplied. The server is not authenticated. This authentication context is typically used when the communication channel offers adequate security to avoid eavesdropping and message (password) replay.

The client has to supply the correct password during the process of AA establishment. If the password is OK, the AA is established and the client can access data within the access rights available in the given AA. Otherwise, the AA is not established.

The LLS authentication process is supported by the COSEM-OPEN service of the COSEM AL – see the complete Green Book – and it is as follows:

- the client transmits a “secret” (for example a password) to the server, by using the “Calling_Authentication_Value” parameter of the COSEM-OPEN.request service primitive;
- the server checks if the “secret” received corresponds to the client identification;
- if yes, the client is authenticated and the AA can be established. If no, the AA is rejected.
- the result with diagnostic information is sent back by the server using the COSEM-OPEN.response service primitive.

The association objects provide a method/attribute to change the “secret”; see DLMS UA 1000-1 4.4.1, 4.4.2.

The authentication process with LLS is shown on the left side of Figure 26.

9.2.3.4 High Level Security (HLS)

The purpose of High Level Security is to allow mutual authentication of the client and the server participating in an association. This authentication context is typically used when the communication channel offers no intrinsic security and precautions have to be taken against eavesdroppers and against message (password) replay.

HLS requires that both the client and the server mutually authenticate each other. This is a 4-pass process, involving the exchange of challenges during AA establishment, which is followed by exchanging the results of processing these challenges, using cryptographic methods. If the authentication takes place, the client can proceed to access data within the access rights available in the given AA, and it accepts data coming from the server. Otherwise, the AA is not established.

The HLS authentication process is supported by the COSEM-OPEN service – see the complete Green Book – and by the Association LN / SN objects as follows:

Pass1: The client transmits a “challenge” CtoS – for example, a random string – to the server;

Pass2: The server transmits a “challenge” StoC – for example, a random string – to the client;

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

Pass3: The client processes StoC according to the rules of the HLS authentication mechanism negotiated:

- in the case of HLS authentication_mechanism_id(2), the method of processing the challenge is secret – for example encrypting with a secret key – which is the HLS secret known by both the client and the server;
- in the case of HLS authentication_mechanism_id(3), the client right concatenates StoC received during Pass2 with the HLS secret, known by both the client and the server, and generates a digest using the MD5 algorithm; see RFC 1321;
- in the case of HLS authentication_mechanism_id(4), the process is the same, but the client generates a digest using the SHA-1 algorithm; see FIPS PUB 180-1;
- in the case of HLS authentication_mechanism_id(5), an authentication tag is calculated using GMAC, see the complete Green Book.

The result – f(StoC) – is sent back 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.

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

The authentication process with HLS is shown on the right side of Figure 26.

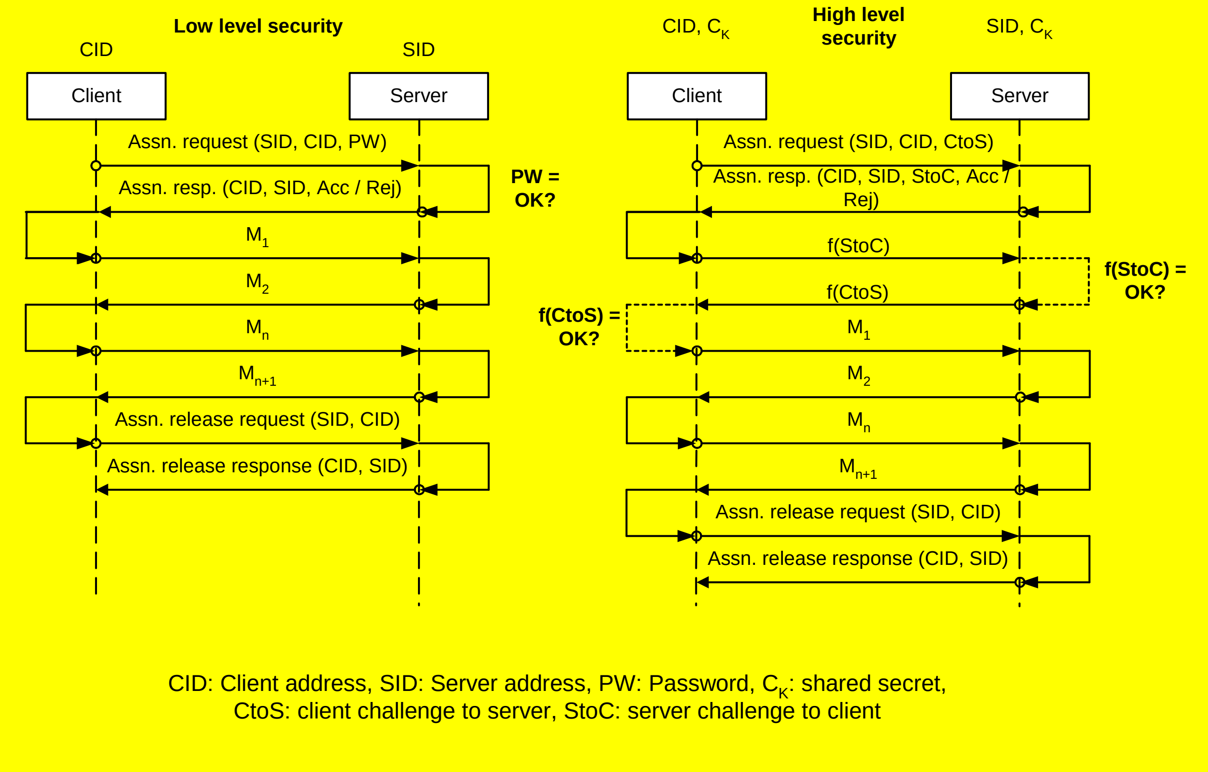

Low level security
CID
SID
Client
Server
Assn. request (SID, CID, PW)
Assn. resp. (CID, SID, Acc / Rej)
PW = OK?
M1
M2
Mn
Mn+1
Assn. release request (SID, CID)
Assn. release response (CID, SID)
High level security
CID, CK
SID, CK
Client
Server
Assn. request (SID, CID, CtoS)
Assn. resp. (CID, SID, StoC, Acc / Rej)
f(StoC)
f(CtoS)
f(StoC) = OK?
f(CtoS) = OK?
M1
M2
Mn
Mn+1
Assn. release request (SID, CID)
Assn. release response (CID, SID)
CID: Client address, SID: Server address, PW: Password, CK: shared secret,
CtoS: client challenge to server, StoC: server challenge to client

Figure 26 – LLS and HLS authentication

Pass1 is supported by the COSEM-OPEN.request service primitive of the client AL. The parameter "Security_Mechanism_Name" carries the identifier of the HLS mechanism, and the parameter "Calling_Authentication_Value" carries the challenge CtoS.

Pass2 is supported by the COSEM-OPEN.response service primitive of the server AL. The parameter "Security_Mechanism_Name" carries the identifier of the HLS mechanism, and the parameter "Responding_Authentication_Value" carries the challenge StoC.

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

Pass3 and Pass4 are supported by the method reply_to_HLS_authentication of the association object(s), see DLMS UA 1000-1 clause 4.4.1 and 4.4.2. If both passes are successfully executed, then full access is granted according to the current AA. Otherwise, either the client or the server aborts the AA.

In addition, the Association SN / LN object provides the method to change the HLS “secret”: change_HLS_secret.

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

9.2.4 Data transport security

9.2.4.1 Applying, removing or checking the protection: ciphering and deciphering

For data transport security, cryptographic protection can be applied before sending an xDLMS APDU – as determined by the security policy, see the complete Green Book – then removed or checked after the reception of an APDU.

NOTE 1 ACSE APDUs are not cryptographically protected, but their user-information field may be cryptographically protected.

NOTE 2 Cryptographic protection of data in storage is out of the scope of this companion specification.

Applying the protection is achieved by ciphering. Deciphering removes or checks the protection, restoring the original xDLMS APDU. Ciphering and deciphering is performed by the COSEM AL as shown in the complete Green Book.

When a COSEM service request or response primitive is invoked by the client or the server AP respectively, the service parameters include the Security_Options parameter. This parameter informs the AL on the requested security to be applied on the xDLMS APDU carrying the service primitive, and it may identify elements of the security material to be used.

Similarly, a COSEM service indication or confirm primitive includes a Security_Status parameter. This parameter informs the AP on the security that was applied on the xDLMS APDU carrying the service primitive, and it may include information on the elements of the security materials used. The authentication tag, if any, is also passed.

For COSEM service definitions, see 9.3.

more details, see complete Green Book ....

9.3 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 27 illustrates this notion of service hierarchy and shows the relationship of the two correspondent N-users and their associated N-layer peer protocol entities.

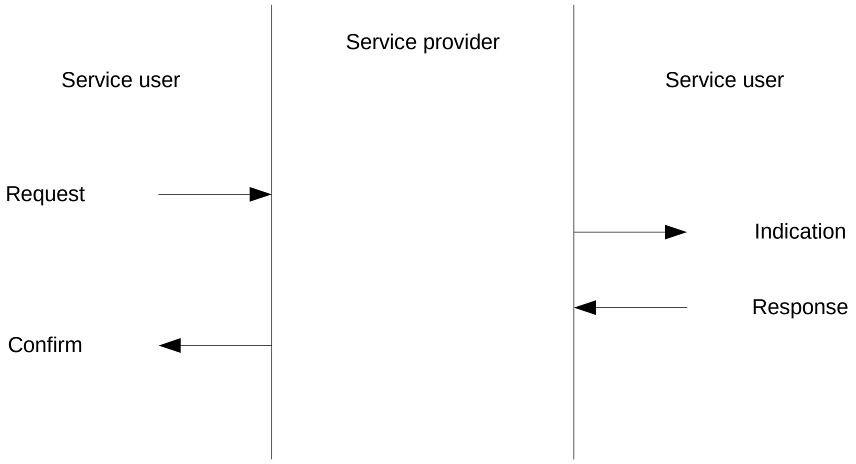


Figure 27 – Service primitives

Services are specified by describing the information flow between the N-user and the N-layer. This information flow is modeled 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 28. 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.

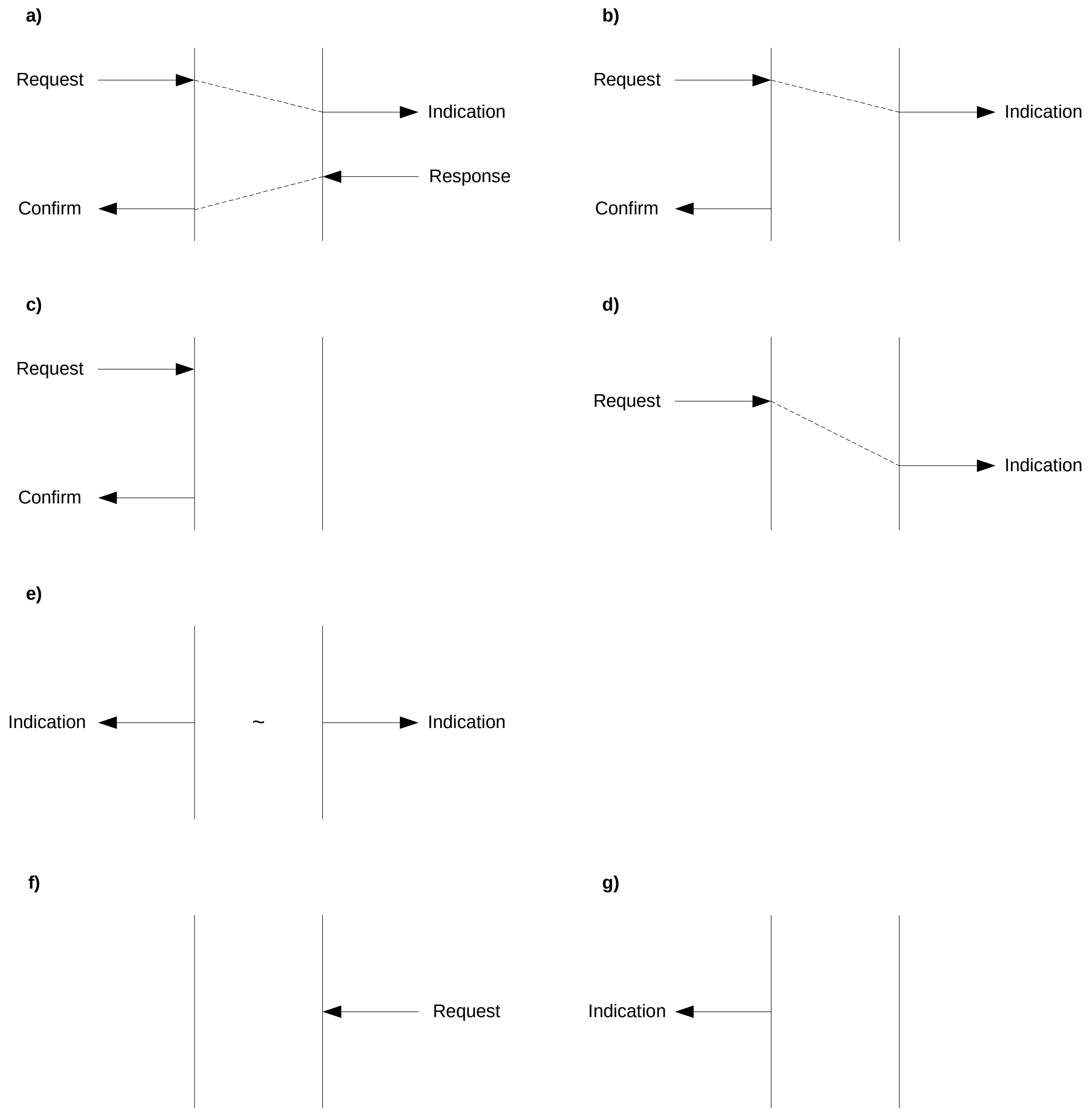


Figure 28 – Time sequence diagrams

The service parameters of the 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 the complete Green Book.

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.

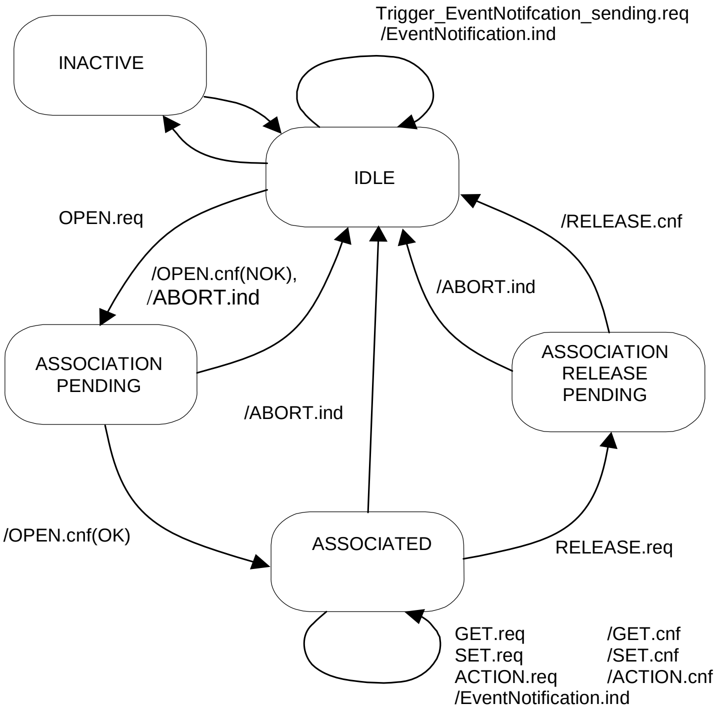
more details, see complete Green Book ....

9.4 COSEM application layer protocol specification

9.4.1 The control function

9.4.1.1 State definitions of the client side control function

Figure 29 shows the state machine for the client side CF, see Figure 24.



NOTE 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 29 – 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 [10]. 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 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.

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. The client/server type data

[10] 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.

services are available in this state, and the EventNotification service can be handled. 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 30 shows the state machine for the server side CF, see Figure 24.

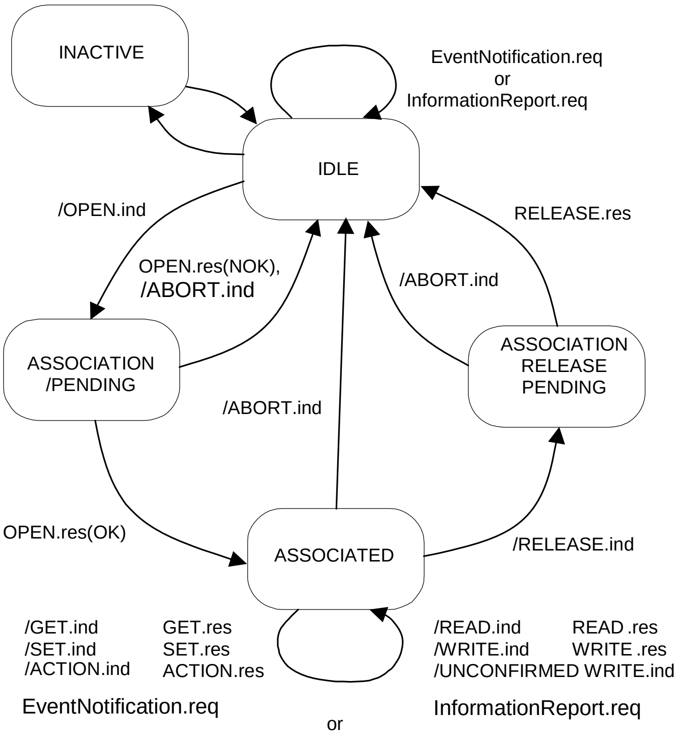


Figure 30 – Partial state machine for the server side control function

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 [10]. 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).

ASSOCIATED — The CF enters this state when the AA has been successfully established. The client/server type data services are available in this state, and the EventNotification / InformationReport services can be handled. 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 COSEM AL ACSE is based on the connection-oriented ACSE, as specified in ISO/IEC 8649 and ISO/IEC 8650-1.

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

- Kernel functional unit;
- Authentication functional unit; and
- Application Context Negotiation functional unit.

The 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. It is the default functional unit. To be included, the Authentication functional unit shall be explicitly requested on the AARQ APDU and accepted on the AARE APDU.

The selection of the Authentication functional unit supports additional parameters on the AARQ and AARE APDUs. The Authentication functional unit does not affect the elements of procedure. Table 5 shows the services and parameters associated with the ACSE functional units, as used by the COSEM AL. The abstract syntax of the ACSE APDUs is specified in the complete Green Book.

Table 5 – ACSE functional units, services and service parameters

Functional Unit	Service	APDU	parameter	Presence
Kernel	A-ASSOCIATE	AARQ	protocol-version	O [1]
			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 [2]	M

Functional Unit	Service	APDU	parameter	Presence
			(carrying an xDLMS-initiate.response APDU) negotiated-quality-of-service negotiated-dlms-version-number negotiated-conformance server-max-receive-pdu-size vaa-name (or carrying a ConfirmedServiceError APDU)	S U M M M M S
	A-RELEASE	RLRQ	reason user-information	U U
		RLRE	reason user-information	U U
Authentication	A-ASSOCIATE	AARQ	sender-acse-requirements mechanism-name calling-authentication-value	U U U
		AARE	responder-acse-requirements mechanism-name responding-authentication-value	U U U

NOTE 1 O: Presence is an ACPM (COSEM Application layer) option.

NOTE 2) According to ISO/IEC 8649, the user-information parameter is optional. However, in the COSEM environment it is mandatory in the AARQ / AARE APDUs.

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

The AARQ and AARE APDU parameters are specified below. Managing these parameters is specified in 9.4.4.1.

- **protocol-version**: the COSEM AL uses the default value. For details, see ISO/IEC 8650-1;
- **application-context-name:** COSEM application context names are specified in 9.4.2.2.2;
- **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 8650-1;
- **implementation-information:** this parameter is not used by the COSEM AL. For details, see ISO/IEC 8650-1;
- **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 pimitive.
- **sender- and responder-acse-requirements:** this parameter 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 module selected;
- **mechanism-name:** COSEM authentication mechanism names are specified in 9.4.2.2.3;
- **calling- and responding- authentication-value:** see 9.2.3;
- **result**: the value of this parameter is determined by the COSEM AP (acceptor) or the 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 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 parameter 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 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 the complete Green Book) is invoked with the parameter Use_RLRQ_RLRE == TRUE – are specified below.

- **reason:** carries the appropriate value as specified in the complete Green Book;
- **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 the complete Green Book.

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 User Association, 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-organisation(5) DLMS-UA(8) }
```

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

- COSEM application context names;
- COSEM authentication mechanism names.

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 [11]. 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-organisation(5) DLMS-UA(8) application-context(1) context_id(x)}
```

The meaning of this general COSEM application contexts is:

[11] 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.

- 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 [12];
- the transfer syntax is A-XDR.

The specific context_ids and the use of ciphered and unciphered APDUs is shown in Table 6 below:

Table 6 – Use of ciphered / unciphered APDUs

Application context name	**context_id**	**Unciphered ADPUs**	**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 authentication (lowest level) security;
- low level, password based authentication security (LLS) identifying only the client;
- high level, four-pass authentication security (HLS) identifying both the client and the server.

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:

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)
NOTE With mechanism_id(2), the method of processing the challenge is secret.	

When the Authentication_Mechanism_Name is present in the COSEM-OPEN service, the authentication functional unit of the A-RELEASE service shall be selected. The process of LLS and HLS authentication is described in 9.2.3 and the complete Green Book.

9.4.3 APDU encoding rules

The ACSE APDUs shall be encoded in BER (ISO/IEC 8825). 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 11 and in 12.

[12] With the COSEM extensions to DLMS, see 9.1.2.3.1.

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 COSEM interoperability. The participants of an AA are:

- a client AP, proposing AAs; and
- a server AP [13], accepting them or not.

Figure 31 gives the MSC for the case, when:

- the COSEM-OPEN.request primitive requests a confirmed AA;
- the connection of the supporting lower 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. 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 [14]. 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.

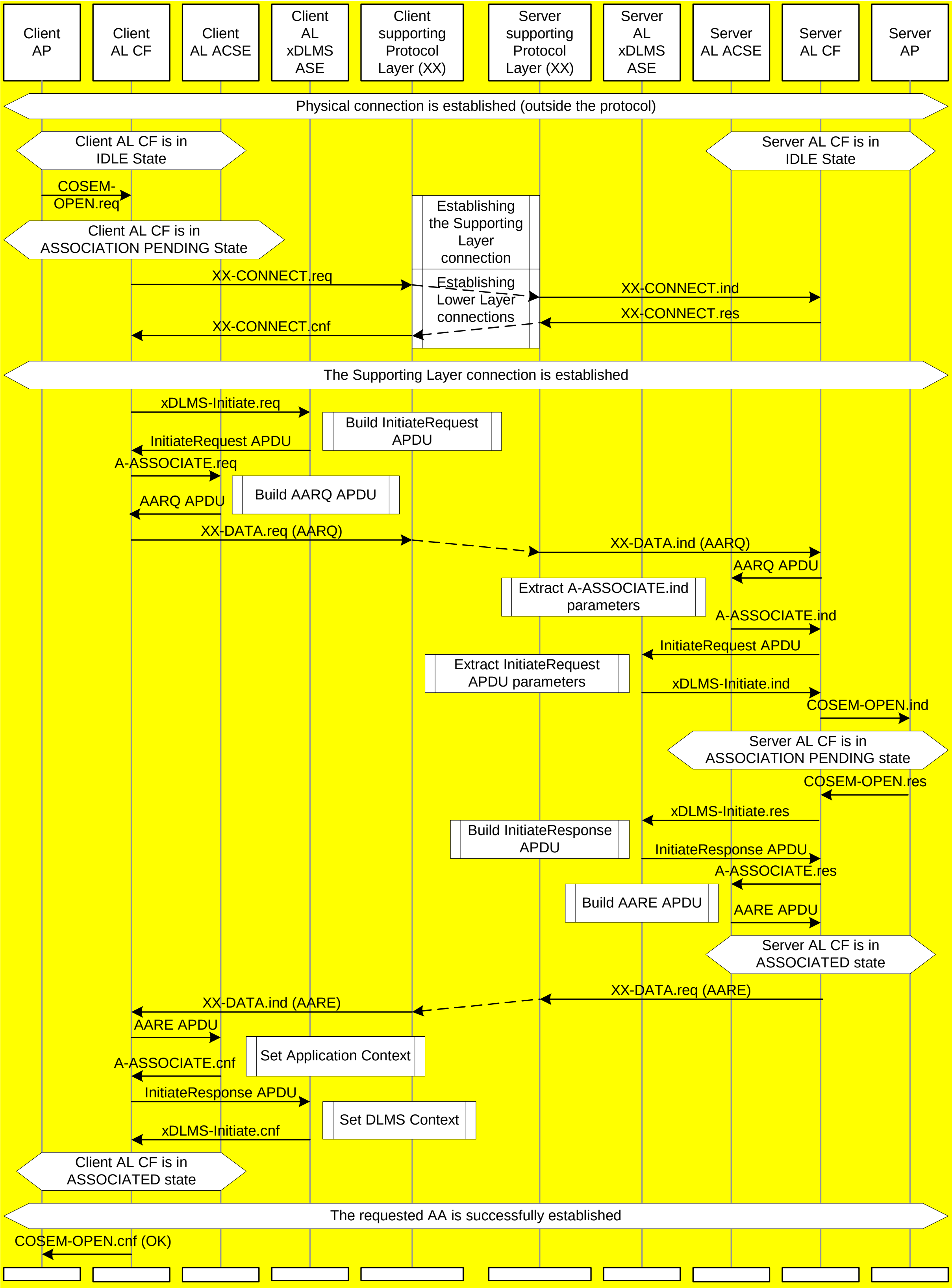
The CF of the server AL gives the AARQ APDU received to the ACSE. It extracts the ACSE related parameters then gives back the control to the CF. The CF passes then the contents of the user-information parameter of the AARQ APDU – carrying an xDLMS InitiateRequest APDU – to the xDLMS_ASE. It retrieves 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 [15] and enters the ‘ASSOCIATION PENDING’ state.

NOTE 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.

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

[14] The PH layer must be connected before the COSEM-OPEN service is invoked.

[15] 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.


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
A-ASSOCIATE.cnf
Set Application Context
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 31 – 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;

  NOTE 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.

Fields of the authentication functional unit (when present):

- sender-acse-requirements:
- if is not present or it is present but bit 0 = 0, then the authentication functional unit is not selected. Any following fields of the authentication functional unit may be ignored;
- 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, the value of this parameter is TRUE (default), the server shall send back an AARE APDU. Otherwise, the server shall not respond. See also 10.
  a) proposed-dlms-version-number, see 9.1.2.3.1;
  b) proposed-conformance, see the complete Green Book.;
  c) client-max-receive-pdu-size, see 9.1.2.3.1.

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;
- Responding-AP-title: if the negotiated application context uses ciphering, it shall carry the server system title.

  NOTE 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.

- Result: accepted;
- Failure_Type: Result-source: acse-service-user; Diagnostic: null;
- User_Information: an xDLMS InitiateResponse APDU carrying the elements of the 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 lower 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 parameters 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;
- User_Information: an xDLMS ConfirmedServiceError APDU, 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 lower layer protocols. The proposed AA is not established, the server CF returns to the IDLE state.

At the client side, the parameters 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 situation, it returns a COSEM-OPEN.response primitive to the client with an appropriate Result parameter. The Result Source parameter is appropriately assigned the symbolic value of “ACSE service-provider”. The COSEM-OPEN.indication primitive is not issued. The association is not established.

more details, see complete Green Book ....

10. Using the COSEM application layer in various communications profiles

10.1 Communication profile specific elements

10.1.1 General

As explained in 4, 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. As shown in Figure 3, in each such profile, the application layer is the 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 COSEM AL services to the service set provided and used by the supporting layer;
- communication profile specific parameters of the 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 scheme

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

As described in DLMS UA 1000-1 Clause 4.1.7, metering equipment are modeled 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 lower 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 need also to identify the AAs.

10.1.5 Supporting layer services and service mapping

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

In each communication profile, the 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 COSEM AL services

In 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 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 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 Clause 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.

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 32.

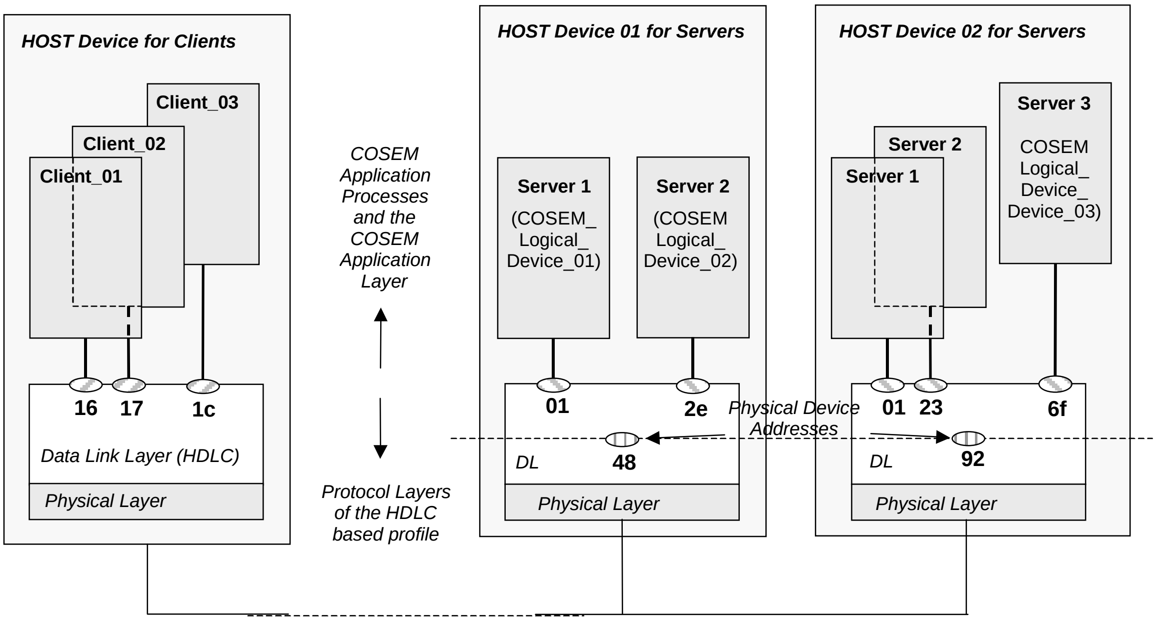

HOST Device for Clients
Client_03
Client_02
Client_01
16
17
1c
Data Link Layer (HDLC)
Physical Layer
COSEM Application Processes and the COSEM Application Layer
Protocol Layers of the HDLC based profile
HOST Device 01 for Servers
Server 1
(COSEM_ Logical_ Device_01)
Server 2
(COSEM Logical_ Device_02)
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 32 – Identification/addressing scheme in the 3-layer, CO, HDLC-based communication profile

10.2.4 Supporting layer services and service mapping

In this profile, the supporting layer of the 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 33 summarizes the data link layer services provided for and used by the 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.

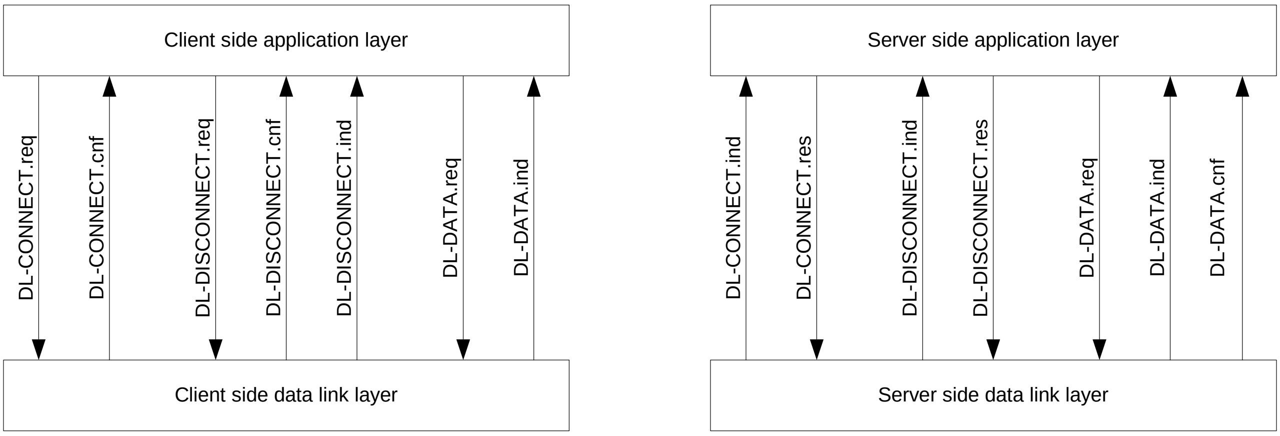


Figure 33 – Summary of data link layer services

10.2.5 Communication profile specific service parameters of the 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 7 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 the complete Green Book – 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, bit 6 of the Invoke-Id-And-Priority field – see the complete Green Book – is not relevant in this profile.

Table 7 – 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

-

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 the complete Green Book) 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 – and COSEM-RELEASE – see the complete Green Book – 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, the following rule applies:

- the Diagnostics parameter of the COSEM-ABORT.indication primitive – see the complete Green Book – 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 companion specification.

10.2.6.4 EventNotification service and protocol

This sub-clause describes the communication profile specific elements of the protocol of the EventNotification service, see the complete Green Book.

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 0x10).

The EventNotificationRequest 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 EventNotificationRequest APDU – following the reception of an I, a UI or an RR frame from the client. See the complete Green Book.

Figure 34 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.

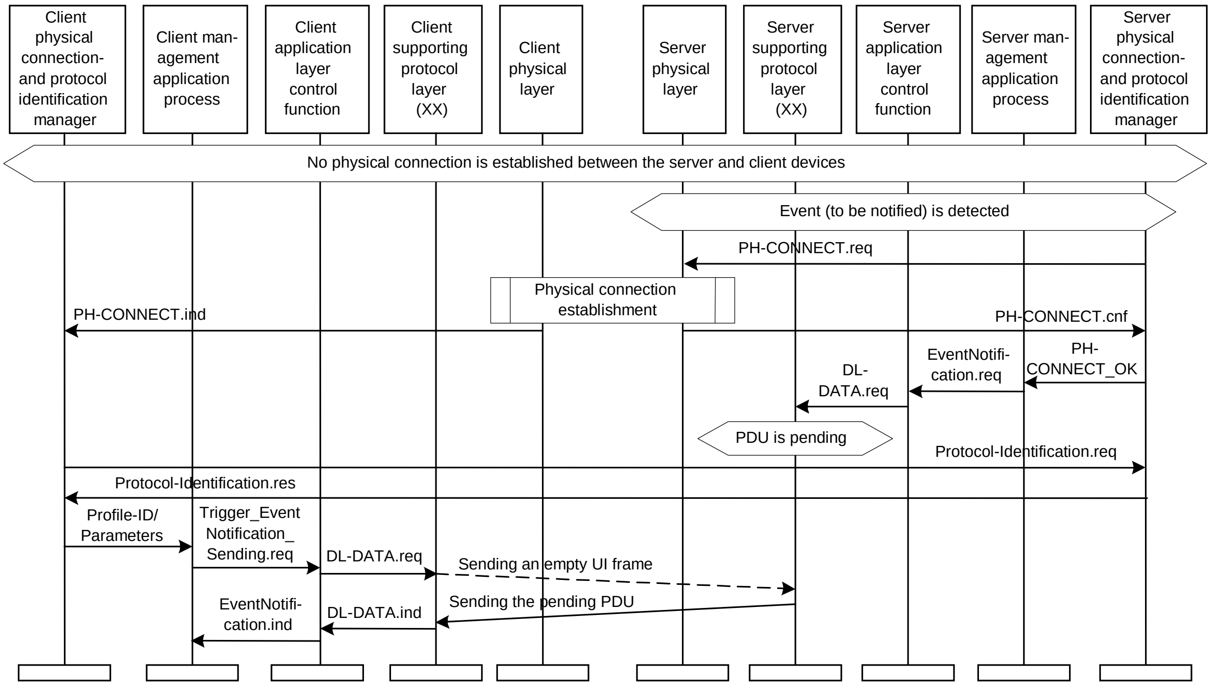

Client physical connection- and protocol identification manager
Client man-agement application process
Client application layer control function
Client supporting protocol layer (XX)
Client physical layer
Server physical layer
Server supporting protocol layer (XX)
Server application layer control function
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 34 – Example: EventNotificaton triggered by the client

The first step is to establish this physical connection [16]. 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 EventNotificationRequestAPDU 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 EventNotificationRequestAPDU.

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 34. Following this, the client is able to instantiate the right protocol stack.

The client AP invokes then the Trigger_EventNotification_Sending .request primitive (see the complete Green Book). 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 EventNotificationRequestAPDU, 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 9.1.2.3.12.

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.

[16] This physical connection establishment is done outside of the protocol stack.

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 modeled with a logical bus, to which the metering equipment and the shared resource are connected, see Figure 35.

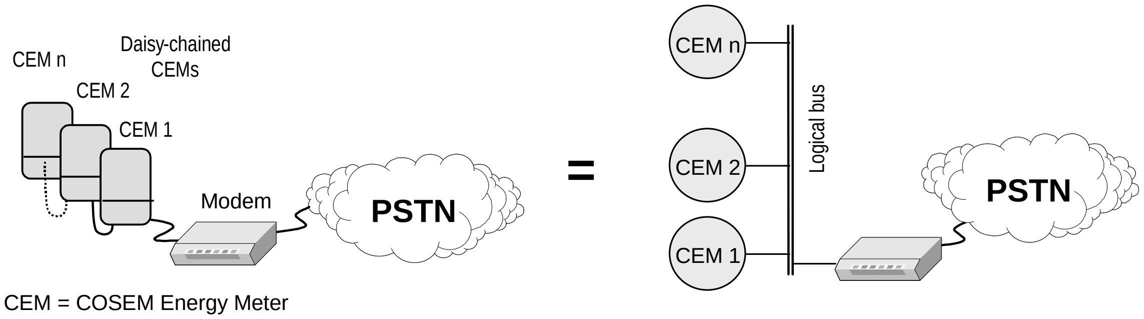


Figure 35 – Multi-drop configuration and its model

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 modeled 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.

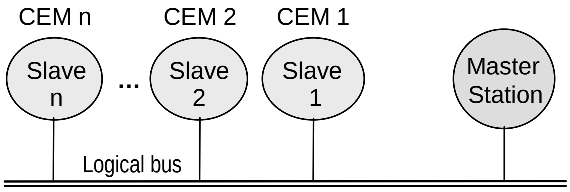


Figure 36 – Master/ Slave operation on the multi-drop bus

The client may send requests at the same time to multiple servers, if no response is expected (multi-cast 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 non-client/server type EventNotification / InformationReport services. As events in several or all meters 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 profiles are suitable for remote data exchange with metering equipment via IP enabled Networks (for example Internet), using various communication networks, such as Local Area Networks or public- or private Wide Area Networks.

10.3.2 The structure of the profile(s)

The COSEM TCP-UDP/IP based communication profiles consist of five protocol layers:

- the COSEM Application layer, specified in Clause 9,
- the COSEM Transport layer, specified in Clause 7;
- a network Layer: the Internet Protocol (IPv4), specified in the Internet standard STD0005;
- a data link layer: any data link protocol supporting IPv4 (PPP, Ethernet, etc.);
- a physical layer: any PhL supported by the data link layer chosen.

The 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 network layer to communicate with other nodes connected to this abstract network. The 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 13.

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.

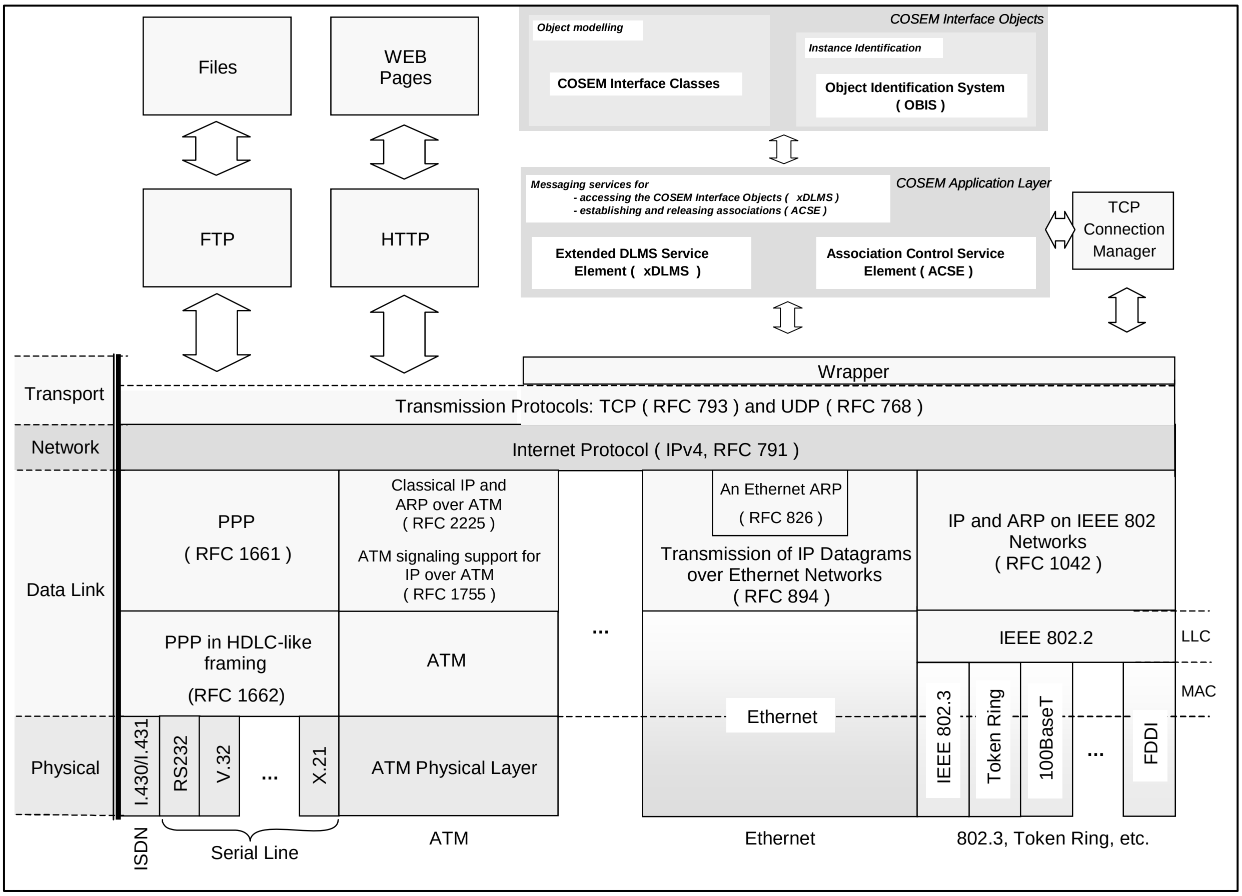

Files
WEB Pages
Object modelling
COSEM Interface Objects
Instance Identification
COSEM Interface Classes
Object Identification System (OBIS)
FTP
HTTP
Messaging services for
- accessing the COSEM Interface Objects (xDLMS)
- establishing and releasing associations (ACSE)
COSEM Application Layer
TCP Connection Manager
Extended DLMS Service Element (xDLMS)
Association Control Service Element (ACSE)
Wrapper
Transport
Transmission Protocols: TCP (RFC 793) and UDP (RFC 768)
Network
Internet Protocol (IPv4, RFC 791)
Data Link
PPP (RFC 1661)
Classical IP and ARP over ATM (RFC 2225)
ATM signaling support for IP over ATM (RFC 1755)
An Ethernet ARP (RFC 826)
Transmission of IP Datagrams over Ethernet Networks (RFC 894)
IP and ARP on IEEE 802 Networks (RFC 1042)
PPP in HDLC-like framing (RFC 1662)
ATM
...
IEEE 802.2
LLC
MAC
Ethernet
Physical
I.430/I.431
RS232
V.32
...
X.21
ATM Physical Layer
IEEE 802.3
Token Ring
100BaseT
...
FDDI
ISDN
Serial Line
ATM
Ethernet
802.3, Token Ring, etc.

Figure 37 – 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), as well as Local Area Networks, such as Ethernet, Wireless LAN, Token Ring, FDDI, Bluetooth, etc. support TCP-UDP/IP networking. Figure 37 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, COSEM can be used practically on any existing communication network.

more details, see complete Green Book ....

11. AARQ and AARE encoding examples

11.1 General

This chapter contains examples of encoding the AARQ and AARE APDUs, in cases of using various levels of authentication and in cases of success and failure.

In COSEM, the AARQ, AARE, RLRQ and RLRE APDUs – see 9.4.2 – shall be encoded in BER (ISO/IEC 8825). 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 ADPUs 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 8.

Table 8 – 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 the following:

Table 9 – A-XDR encoding the xDLMS InitiateRequest APDU

-- A-XDR encoding the xDLM InitiateRequest APDU	**LN referencing**	**SN referencing**
// encoding of the tag of the DLMS 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 xDLSM InitiateResponse APDU are encoded in BER. That's why the length of the bit-string and the number of the unused bits areencoded.

[2] For encoding of identifier octets, see ISO/IEC 8825 Ed.3.0:2002, Clause 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 the following:

Table 10 – A-XDR encoding the xDLMS InitiateResponse APDU

-- A-XDR encoding the xDLMS InitiateResponse APDU	**LN referencing**	**SN referencing**
// encoding of the tag of the DLMS 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

more details, see complete Green Book ....

12. Encoding examples: AARQ and AARE APDUs using a ciphered application context

NOTE The System Title is the same in each example. In the reality, the System Title in the request and in the response APDUs should be different, as they are originated by different systems.

12.1 A-XDR encoding of the xDLMS InitiateRequest APDU, carrying a dedicated key

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 1200 bytes (0x04B0).

Table 11 – A-XDR encoding of the xDLMS InitiateRequest APDU

// encoding of the tag of the DLMS 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, Annex C, Examples 1 and 2, the proposed-conformance element of the xDLMS InitiateRequest APDU and the negotiated-conformance element of the xDLSM 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 Ed.3.0:2002, Clause 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.

more details, see 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 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.

more details, see complete Green Book ....

14. Data transfer service examples

The following tables show examples for data exchange using xDLMS services using LN referencing (left column) and SN referencing (right column).

Table 12 – 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
-   0102030405060708091011121314151 6
-   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.

more details, see complete Green Book ....

15. Overview of cryptography (Informative)

NOTE This clause is provided as a brief introduction to the cryptography tools selected. More information can be found in the documents referenced.

15.1 General

NOTE The following text is quoted from NIST SP 800-21 3.1.

Cryptography is a branch of mathematics that is based on the transformation of data and can be used to provide several security services: confidentiality, data integrity, authentication, authorization and non-repudiation. Cryptography relies upon two basic components: an *algorithm* (or cryptographic methodology) and a *key*. The algorithm is a mathematical function, and the key is a parameter used in the transformation.

A cryptographic algorithm and key are used to apply cryptographic protection to data (e.g., encrypt the data or generate a digital signature) and to remove or check the protection (e.g., decrypt the encrypted data or verify the digital signature). There are three basic types of approved cryptographic algorithms: cryptographic hash functions, symmetric key algorithms and asymmetric key algorithms:

- cryptographic hash functions do not require keys (although they can be used in a mode in which keys are used). A hash function is often used as a component of an algorithm to provide a security service. See 15.2;
- symmetric algorithms (often called secret key algorithms) use a single key to both apply the protection and to remove or check the protection. See 15.3;
- asymmetric algorithms (often called public key algorithms) use two keys (i.e., a key pair): a public key and a private key that are mathematically related to each other, see 15.4.

In order to use cryptography, cryptographic keys must be "in place", i.e., keys must be established for parties using cryptography.

15.2 Hash functions

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.2.

A hash function produces a short representation of a longer message. A good hash function is a one-way function: it is easy to compute the hash value from a particular input; however, backing up the process from the hash value back to the input is extremely difficult. With a good hash function, it is also extremely difficult to find two specific inputs that produce the same hash value. Because of these characteristics, hash functions are often used to determine whether or not data has changed.

A hash function takes an input of arbitrary length and outputs a fixed length value. Common names for the output of a hash function include *hash value* and *message digest*. Figure 38 depicts the use of a hash function.

A hash value (H1) is computed on data (M1). M1 and H1 are then saved or transmitted. At a later time, the correctness of the retrieved or received data is checked by labelling the received data as M2 (rather than M1) and computing a new hash value (H2) on the received value. If the newly computed hash value (H2) is equal to the retrieved or received hash value (H1), then it can be assumed that the retrieved or received data (M2) is the same as the original data (M1) (i.e., M1 = M2).

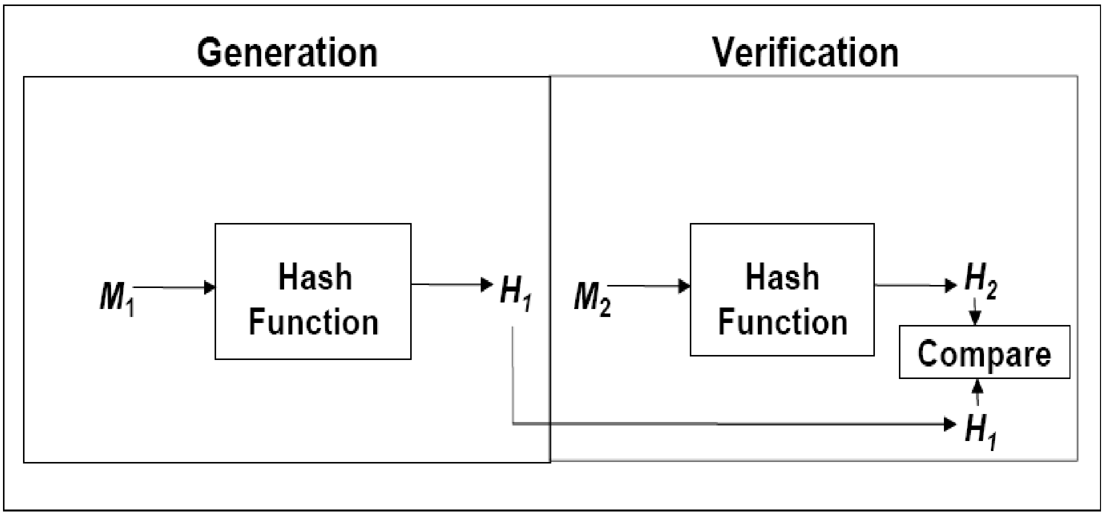


Figure 38 – Hash function

15.3 Symmetric key algorithms

15.3.1 General

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.3.

Symmetric key algorithms (often called secret key algorithms) use a single key to both apply the protection and to remove or check the protection. For example, the key used to encrypt data is also used to decrypt the encrypted data. This key must be kept secret if the data is to retain its cryptographic protection. Symmetric algorithms are used to provide confidentiality via encryption, or an assurance of authenticity or integrity via authentication, or are used during key establishment.

Keys used for one purpose shall not be used for other purposes. (See NIST SP 800-57).

15.3.2 Encryption and decryption

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.3.1.

Encryption is used to provide confidentiality for data. The data to be protected is called *plaintext*. Encryption transforms the data into *ciphertext*. Ciphertext can be transformed back into plaintext using decryption. The approved algorithms for encryption and decryption algorithms are: the Advanced Encryption Standard (AES) and the Triple Data Encryption Algorithm (TDEA). TDEA is based on the Data Encryption Standard (DES), which is no longer approved for Federal Government use except as a component of TDEA. Each of these algorithms operates on blocks (chunks) of data during an encryption or decryption operation. For this reason, these algorithms are commonly referred to as block cipher algorithms.

Plaintext data can be recovered from ciphertext only by using the same key that was used to encrypt the data. Unauthorized recipients of the ciphertext who know the cryptographic algorithm but do not have the correct key should not be able to decrypt the ciphertext. However, anyone who has the key and the cryptographic algorithm can easily decrypt the ciphertext and obtain the original plaintext data.

Figure 39 depicts the encryption and decryption processes. The plaintext (P) and a key (K) are used by the encryption process to produce the ciphertext (C). To decrypt, the ciphertext (C) and the same key (K) are used by the decryption process to recover the plaintext (P).

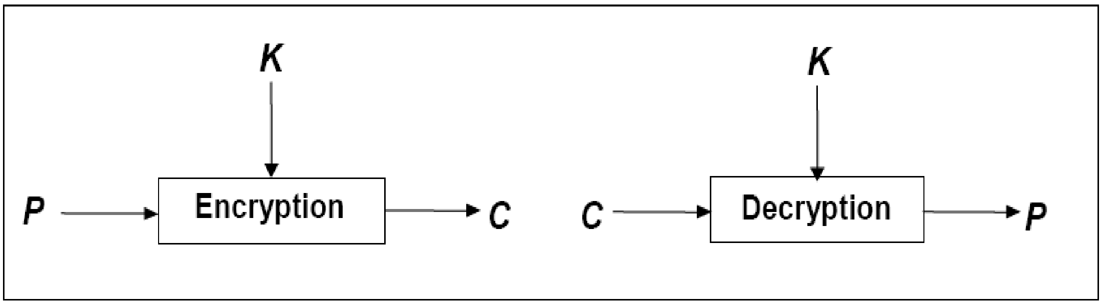


Figure 39 – Encryption and decryption

With symmetric key block cipher algorithms, the same plaintext block and key will always produce the same ciphertext block. This property does not provide acceptable security. Therefore, cryptographic modes of operation have been defined to address this problem (see 15.3.4).

15.3.3 Advanced Encryption Standard (AES)

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.3.1.3.

The Advanced Encryption Standard (AES) was developed as a replacement for DES and is the preferred algorithm for new products. AES is specified in FIPS PUB 197. AES encrypts and decrypts data in 128-bit blocks, using 128, 192 or 256 bit keys. All three key sizes are adequate.

NOTE The following text is quoted from RFC 5084.

AES offers a combination of security, performance, efficiency, ease of implementation, and flexibility. Specifically, the algorithm performs well in both hardware and software across a wide range of computing environments. Also, the very low memory requirements of the algorithm make it very well suited for restricted-space environments.

15.3.4 Encryption Modes of Operation

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.3.1.4.

With a symmetric key block cipher algorithm, the same plaintext block will always encrypt to the same ciphertext block when the same symmetric key is used. If the multiple blocks in a typical message (data stream) are encrypted separately, an adversary could easily substitute individual blocks, possibly without detection. Furthermore, certain kinds of data patterns in the plaintext, such as repeated blocks, would be apparent in the ciphertext.

Cryptographic modes of operation have been defined to address this problem by combining the basic cryptographic algorithm with variable initialization values (commonly known as initialization vectors) and feedback rules for the information derived from the cryptographic operation.

NIST SP 800-38D specifies the Galois/Counter Mode (GCM), an algorithm for authenticated encryption with associated data, and its specialization, GMAC, for generating a message authentication code (MAC) on data that is not encrypted. GCM and GMAC are modes of operation for an underlying approved symmetric key block cipher. See the complete Green Book.

15.3.5 Message Authentication Code

15.3.5.1 General

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.3.2.

Message Authentication Codes (MACs) provide an assurance of authenticity and integrity. A MAC is a cryptographic checksum on the data that is used to provide assurance that the data has not changed or been altered and that the MAC was computed by the expected party (the sender). Typically, MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.

Figure 40 depicts the use of message authentication codes (MACs).

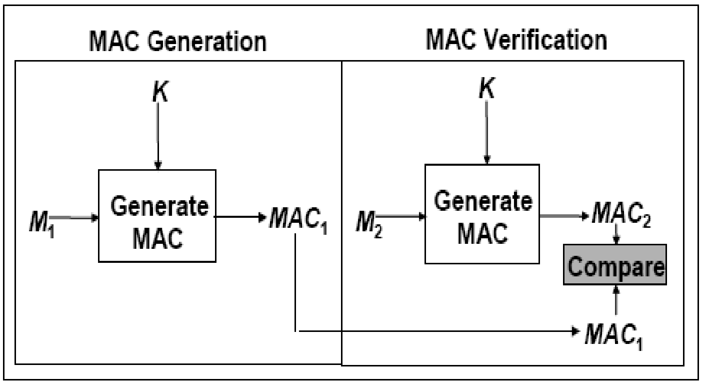


Figure 40 – Message Authentication Codes (MACs)

A MAC (MAC1) is computed on data (M1) using a key (K). M1 and MAC1 are then saved or transmitted. At a later time, the authenticity of the retrieved or received data is checked by labeling the retrieved or received data as M2 and computing a MAC (MAC2) on it using the same key (K). If the retrieved or received MAC (MAC1) is the same as the newly computed MAC (MAC2), then it can be assumed that the retrieved or received data (M2) is the same as the original data (M1) (i.e., M1 = M2). The verifying party also knows who the sending party is because no one else knows the key.

Typically, MACs are used to detect data modifications that occur between the initial generation of the MAC and the verification of the received MAC. They do not detect errors that occur before the MAC is originally generated.

Message integrity is frequently provided using non-cryptographic techniques known as error detection codes. However, these codes can be altered by an adversary to the adversary's benefit. The use of an approved cryptographic mechanism, such as a MAC, addresses this problem. That is, the integrity provided by a MAC is based on the assumption that it is not possible to generate a MAC without knowing the cryptographic key. An adversary without knowledge of the key will be unable to modify data and then generate an authentic MAC on the modified data. It is therefore crucial that MAC keys be kept secret.

Two types of algorithms for computing a MAC have been approved for Federal government use: MAC algorithms that are based on block cipher algorithms, and MAC algorithms that are based on hash functions.

15.3.5.2 The Keyed-Hash Message Authentication Code (HMAC)

NOTE The following text is quoted from FIPS PUB 198.

The Keyed-Hash Message Authentication Code (HMAC) uses a cryptographic hash function in conjunction with a secret key. HMAC shall be used in combination with an approved cryptographic hash function. HMAC uses a secret key for the calculation and verification of the MACs. For details, see FIPS PUB 198.

15.3.6 Key establishment

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.3.3.

Symmetric key algorithms may be used to wrap (i.e., encrypt) keying material using a key-wrapping key (also known as a key encrypting key). The wrapped keying material can then be stored or transmitted securely. Unwrapping the keying material requires the use of the same key-wrapping key that was used during the original wrapping process.

Key wrapping differs from simple encryption in that the wrapping process includes an integrity feature. During the unwrapping process, this integrity feature detects accidental or intentional modifications to the wrapped keying material.

15.4 Asymmetric key algorithms

15.4.1 Introduction

The use of asymmetric key algorithms for DLMS/COSEM is under consideration.

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.4.

Asymmetric algorithms (often called public key algorithms) use two keys: a public key and a private key, which are mathematically related to each other. The public key may be made public; the private key must remain secret if the data is to retain its cryptographic protection. Even though there is a relationship between the two keys, the private key cannot be determined from the public key. Which key to be used to apply versus remove or check the protection depends on the service to be provided. For example, a digital signature is computed using a private key, and the signature is verified using the public key; for those algorithms also capable of encryption [17], the encryption is performed using the public key, and the decryption is performed using the private key.

Asymmetric algorithms are used primarily as data integrity, authentication, and non-repudiation mechanisms (i.e., digital signatures), and for key establishment.

Some asymmetric algorithms use domain parameters, which are additional values necessary for the operation of the cryptographic algorithm. These values are mathematically related to each other. Domain parameters are usually public and are used by a community of users for a substantial period of time.

The secure use of asymmetric algorithms requires that users obtain certain assurances:

- assurance of domain parameter validity provides confidence that the domain parameters are mathematically correct;
- assurance of public key validity provides confidence that the public key appears to be a suitable key; and
- assurance of private key possession provides confidence that the party that is supposedly the owner of the private key really has the key.

Some asymmetric algorithms may be used for multiple purposes (e.g., for both digital signatures and key establishment). Keys used for one purpose shall not be used for other purposes.

15.4.2 Digital signatures

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.4.1.

A digital signature is an electronic analogue of a written signature that can be used in proving to the recipient or a third party that the message was signed by the originator (a property known as non-repudiation). Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at a later time.

Digital signatures authenticate the integrity of the signed data and the identity of the signatory. A digital signature is represented in a computer as a string of bits and is computed using a digital signature algorithm that provides the capability to generate and verify signatures. Signature generation uses a private key to generate a digital signature. Signature verification uses the public key that corresponds to, but is not the same as, the private key to verify the signature. Each signatory possesses a private and public key pair. Signature generation can be performed only by the possessor of the signatory's private key. However, anyone can verify the signature by employing the signatory's public key. The security of a digital signature system is dependent on maintaining the secrecy of a signatory’s private key. Therefore, users must guard against the unauthorized acquisition of their private keys.

[17] Not all public key algorithms are capable of multiple functions, e.g., generating digital signatures and encryption.

15.4.3 Key establishment

NOTE The following text is quoted from NIST SP 800-21 sub-clause 3.4.2.

Two types of asymmetric key (i.e., public key) establishment are defined: *key transport* and *key agreement*. Approved key establishment schemes are specified in NIST SP 800-56, *Recommendation on Key Establishment Schemes*.

Key transport is the distribution of a key (and other keying material) from one party to another party. The transported key is created by the sending party. The keying material is encrypted by the sending party and decrypted by the receiving party. The sending party encrypts the keying material using the receiving party's public key; the receiving party decrypts the received keying material using the associated private key.

Key agreement is the participation by both parties (i.e., the sending and receiving parties) in the creation of shared keying material. Each party has either one or two key pairs[18], and the public keys are made known to the other party. The key pairs are used to compute a shared secret[19] value, which is then used with other information to derive keying material using a key derivation function. Typically, a hash function (see 15.2) is used during the key derivation process.

[18] A key pair consists of a private key and its associated public key.

[19] The shared secret is never transmitted from one party to the other.