

Green Book

A series of horizontal lines in various shades of blue and white, extending from the left edge of the slide and ending on the right side, creating a modern, layered effect.

Agenda

- Terms And Definition
- Name & Addressing
- Application Association
- Application Model
- Interconnectivity and Interoperability



Terms & Definitions

Terms And Definitions

➤ Physical Device

- The Physical device, is our meter, it supports one or more communication profiles.

➤ Logical Device

- A physical device hosts one or several Logical Devices.
- A logical device models a specific functionality of the physical device.
- For example, in a multi-energy meter, one logical device could be an electricity meter, another, a gas-meter etc.
- Each logical device has an address, called the logical device address.

➤ Application Entity

- The system-independent application activities that are made available as application services to the application agent.

➤ Application Association

- A cooperative relationship between two application entities, formed by their exchange of application protocol control information through their use of presentation services.

➤ Application Process

- performs the information processing for a particular application.

➤ Application Context

- Set of application service elements, related options and any other information necessary for the interworking of application entities in an application association.

➤ Cosem Data

- COSEM interface object attribute values, method invocation and return parameters.

➤ Cosem Interface Class

- An entity with specific set of attributes and methods modelling a certain function on its own or in relation with other interface classes.

➤ Cosem Interface Object

- An instance of the Cosem Interface Class.

➤ DLMS Message

- An xDLMS APDU.

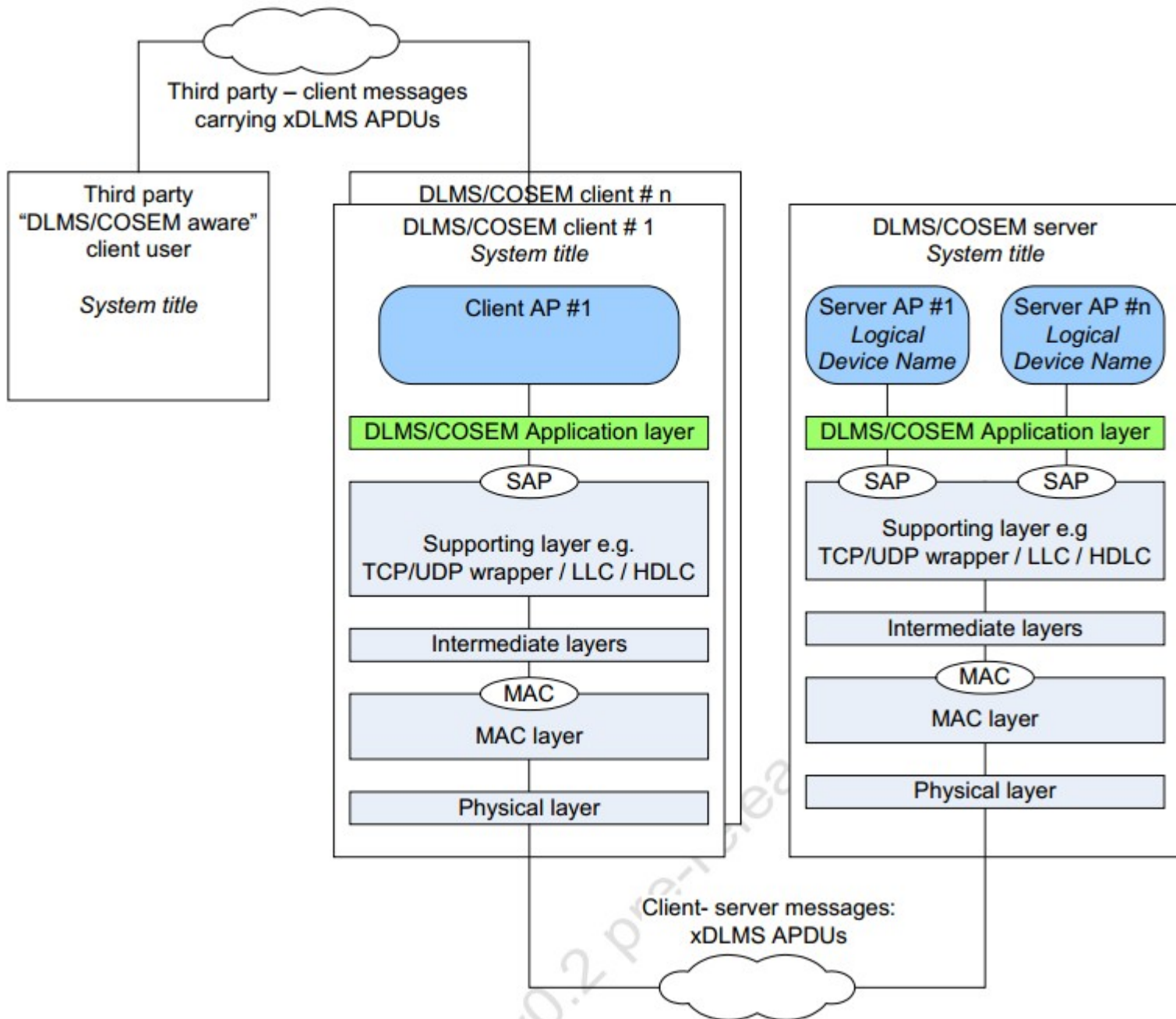
➤ PULL Operation

- a style of communication where the request for a given transaction is initiated by the client

➤ PUSH Operation

- a style of communication where the request for a given transaction is initiated by the server

Naming And Addressing



Naming And Addressing

➤ Naming

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

➤ Address

- Each physical device shall have an appropriate address.
- It depends on the communication profile and may be a phone number, a MAC address, an IP network address or a combination of these.
- Physical device addresses may be pre-configured or may be assigned during a registration process, which also involves binding between the addresses and the system titles.
- Each COSEM client and COSEM logical device AE is bound to a Service Access Point (SAP).
- The SAPs reside in the supporting layer of the COSEM AL.

Client SAPs

No-station	0x00
Client Management Process / CIASE ¹	0x01
Public client	0x10
<i>Open for client AP assignment</i>	0x02 ...0x0F 0x11 and up

Server SAPs

No-station / CIASE ¹	0x00
Management Logical Device	0x01
Reserved for future use	0x02...0x0F
<i>Open for server SAP assignment</i>	0x10 and up
All-station (Broadcast)	Communication profile specific ²

➤ System Title

- shall be 8 octets long; shall be unique.
- The leading (i.e., the 3 leftmost) octets should hold the three-letter manufacturer ID1, the remaining 5 octets shall ensure uniqueness.
- The use of system title in cryptographic protection of xDLMS messages and COSEM data .

➤ Logical Device Name

- The LDN is defined as an octet-string of up to 16 octets. The first three octets shall carry the manufacturer identifier 1.
- The manufacturer shall ensure that the LDN, starting with the three octets identifying the manufacturer and followed by up to 13 octets, is unique.

➤ Client/Server Operation

➤ Connection Oriented Operation

EHL	Secure Meters Limited
EIT	EnergyICT NV, 8500 Kortrijk, Belgium
EKA	Eka Systems, Germantown, MD 20874, USA
EKO	EKOLIS, PA de Beaujardin Bat Redhae, Chateaugiron, France
EKT	PA KVANT J.S., Russian Federation
ELD	Elektromed Elektronik Ltd, Turkey, O.S.B. Uygurlar Cad. No:4 Sincan, Ankara, Turkey
ELE	Elster Electricity LLC, 208 Rogers Lane, Raleigh, USA

Client User Identification

- This new feature enables the server to distinguish between different users from the client side and to log their activities accessing the meter.
- Each AA established between a client and a server can be used by several users on the client side.
- The properties of the AA are configured in the server, using the “Association” and the “Security set up” objects.
- The list of users – identified by their user_id and user_name – is known both by the client and the server.
- In the server it is held by the user_list attribute of the “Association” objects.
- During AA establishment, the user_id – belonging to the user_name – is carried by the calling-AE-invocation-id field of the AARQ APDU.
- If the user_id provided is on the user_list, the AA can be established – provided that all other conditions are met – and the current_user attribute is updated.
- The value of this attribute can be logged

Application Association

Application Associations

- Application Associations (AAs) are logical connections between a client and a server AE.
- Each AA determines the contexts in which information exchange takes place.
- A COSEM logical device may support one or more AAs, each with a different client.
- AAs may be confirmed or unconfirmed, they may be established on the request of a client ,or may be pre-established .
- A confirmed AA is proposed by the client and accepted by the server provided.
 - The elements of the proposed xDLMS context can be successfully negotiated.

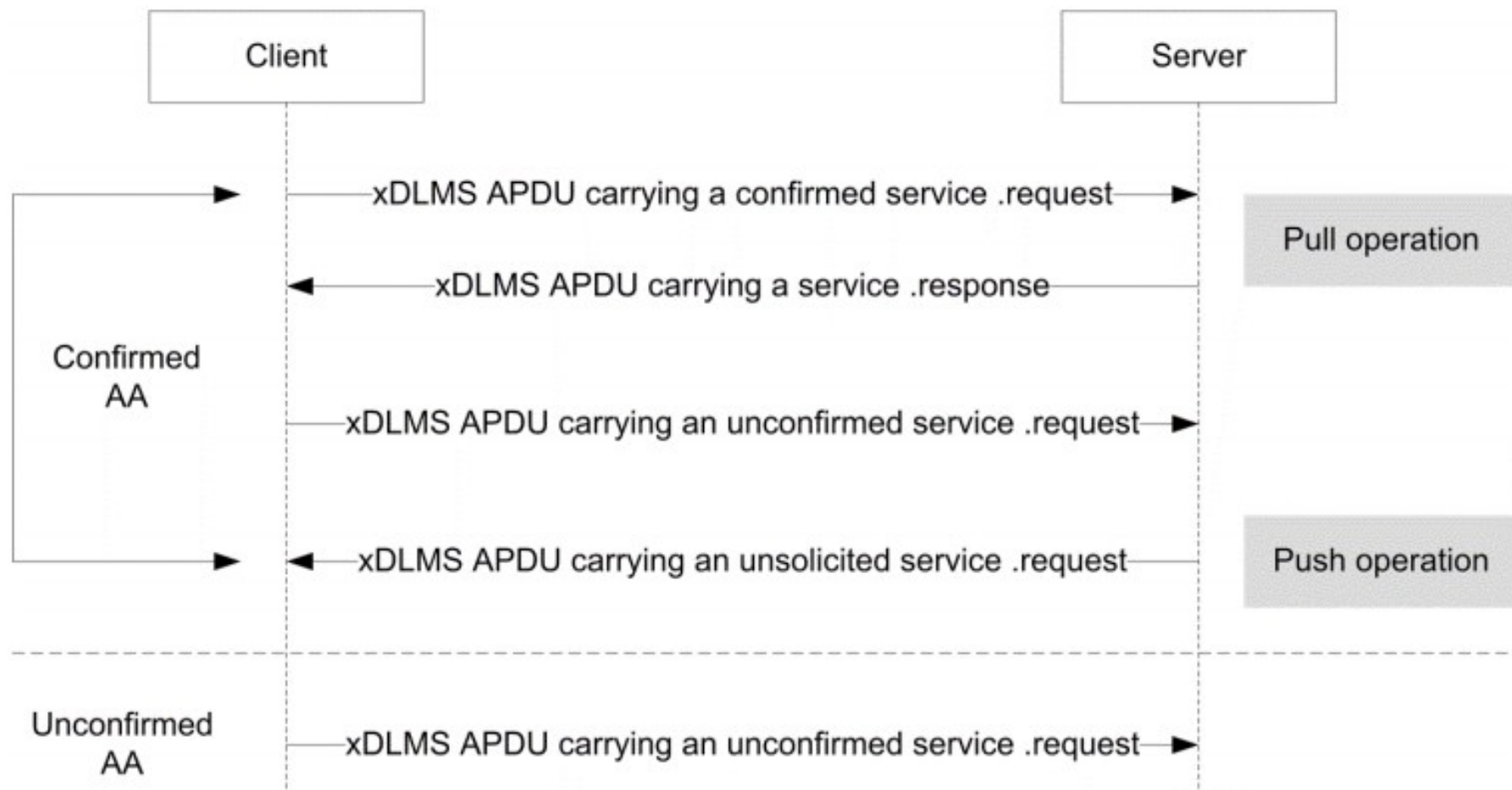
- An unconfirmed AA is also proposed by a client with the assumption that the server will accept it.
 - No negotiation takes place.
 - Unconfirmed AAs are useful for sending broadcast messages from the client to servers.
- An AA can be established only if the client and the server successfully authenticate themselves.
- DLMS/COSEM information exchange can take place only within established AAs.
- AAs are modelled by COSEM “Association SN/ LN” objects that hold the SAPs identifying the associated partners, the name of the application context, the name of the authentication mechanism and the xDLMS context.

➤ The Sub-parts of the Application Association

- Application Context
- Authentication
- xDLMS context
- Security context
 - ✓ The security context is relevant when the application context stipulates ciphering.
- Access Rights
- Messaging Patterns .

➤ Communication Profile

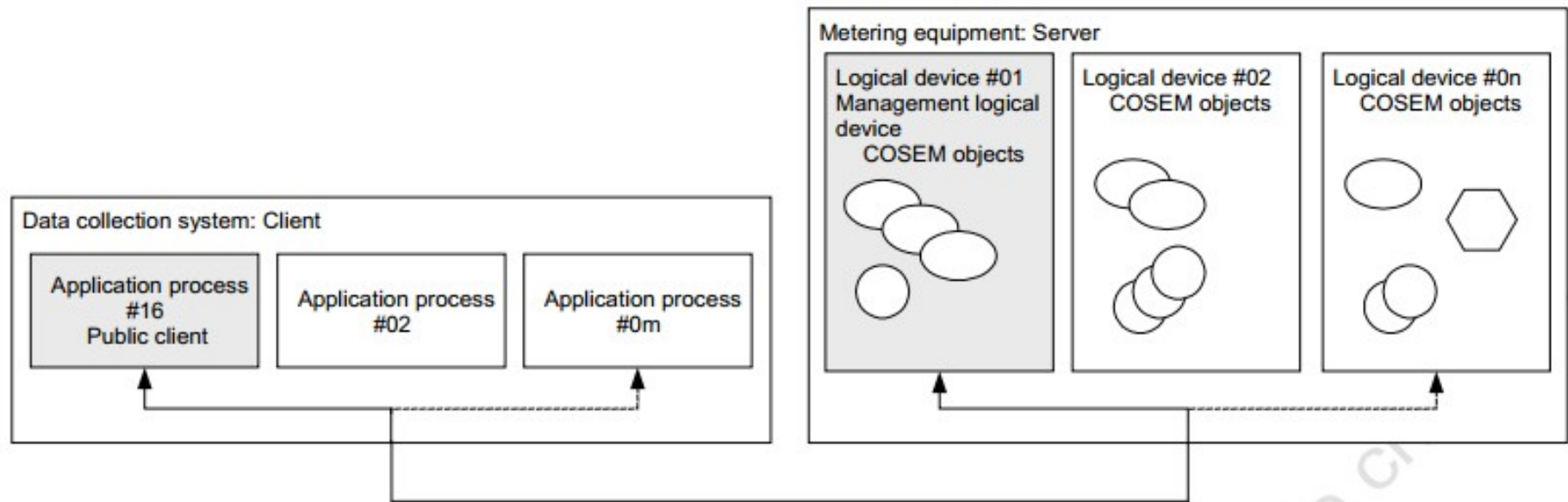
➤ Data Exchange between Third Parties and DLMS/COSEM servers.



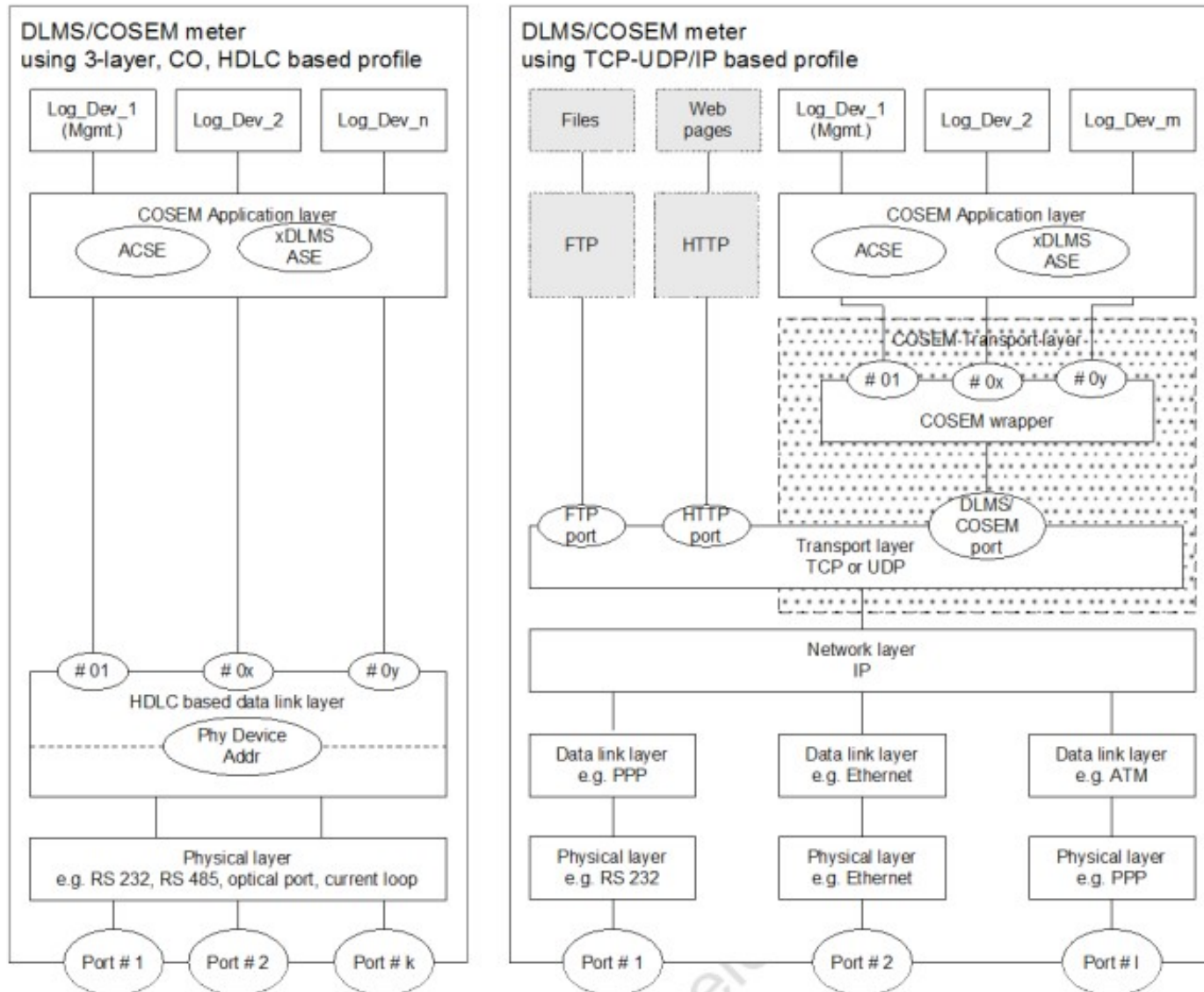


Application Models

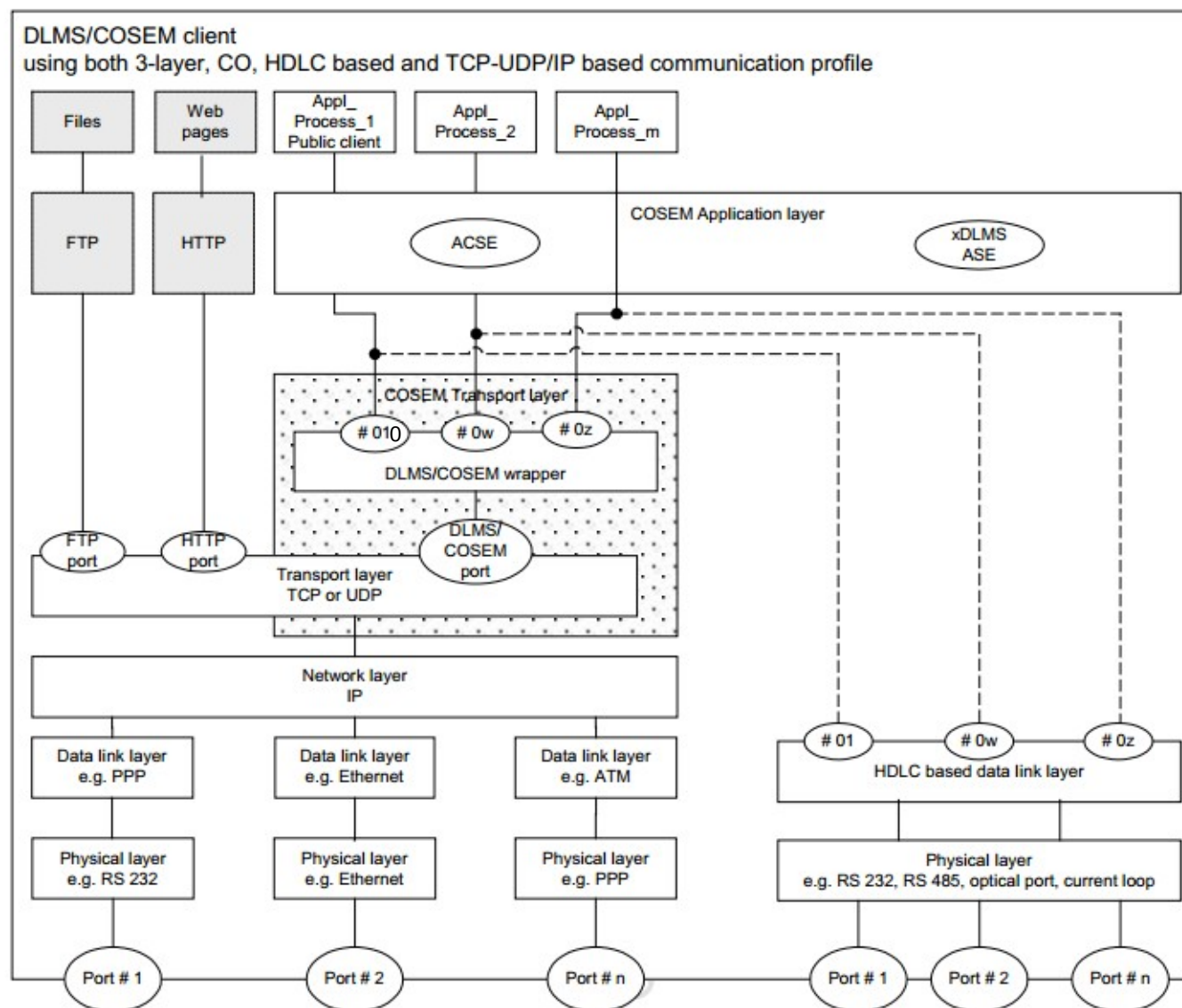
Application Models



Model of a DLMS/COSEM based Server



Model of a DLMS/COSEM based Client



Interoperability & Inter-connectivity in COSEM

- Interoperability and Interconnectivity is defined between client and server AEs.
- Interoperability : a client AE is interoperable with a server AE, if it is able to establish AAs using the A-Associate service of the standard connection-oriented ACSE.
 - AAs may be established between a client AE and server AEs using various application contexts, authentication mechanisms and xDLMS contexts as well as other parameters.
 - For example, a client AE may establish an AA with a server AE with an application context using short name (SN) referencing and with another server AE with an application context using logical name (LN) referencing.
 - Although the messages exchanged depend on the application context of the AA established, both server AEs are interoperable with the client AE 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 DLMS/COSEM.

- **Interconnectivity** : in order to be able to exchange messages, the client and the server AEs should be interconnectable and interconnected.
 - Before the two AEs can establish an AA, they must be interconnected.
 - The two AEs are interconnected, if each peer protocol layer of both sides, which needs to be connected, is connected.
 - In order to be interconnected, the client and server AEs should be interconnectable and shall establish the required connections. Two AEs are interconnectable if they use the same communication profile.
 - With this, interconnectivity in DLMS/COSEM is ensured by the ability of the DLMS/COSEM AE to establish a connection between all peer layers, which need to be connected.
- **Ensuring interconnectivity: the protocol identification service**
 - In DLMS/COSEM, AA establishment is always initiated by the client AE.
 - However, in some cases, it may not have knowledge about the protocol stack used by an unknown server device .

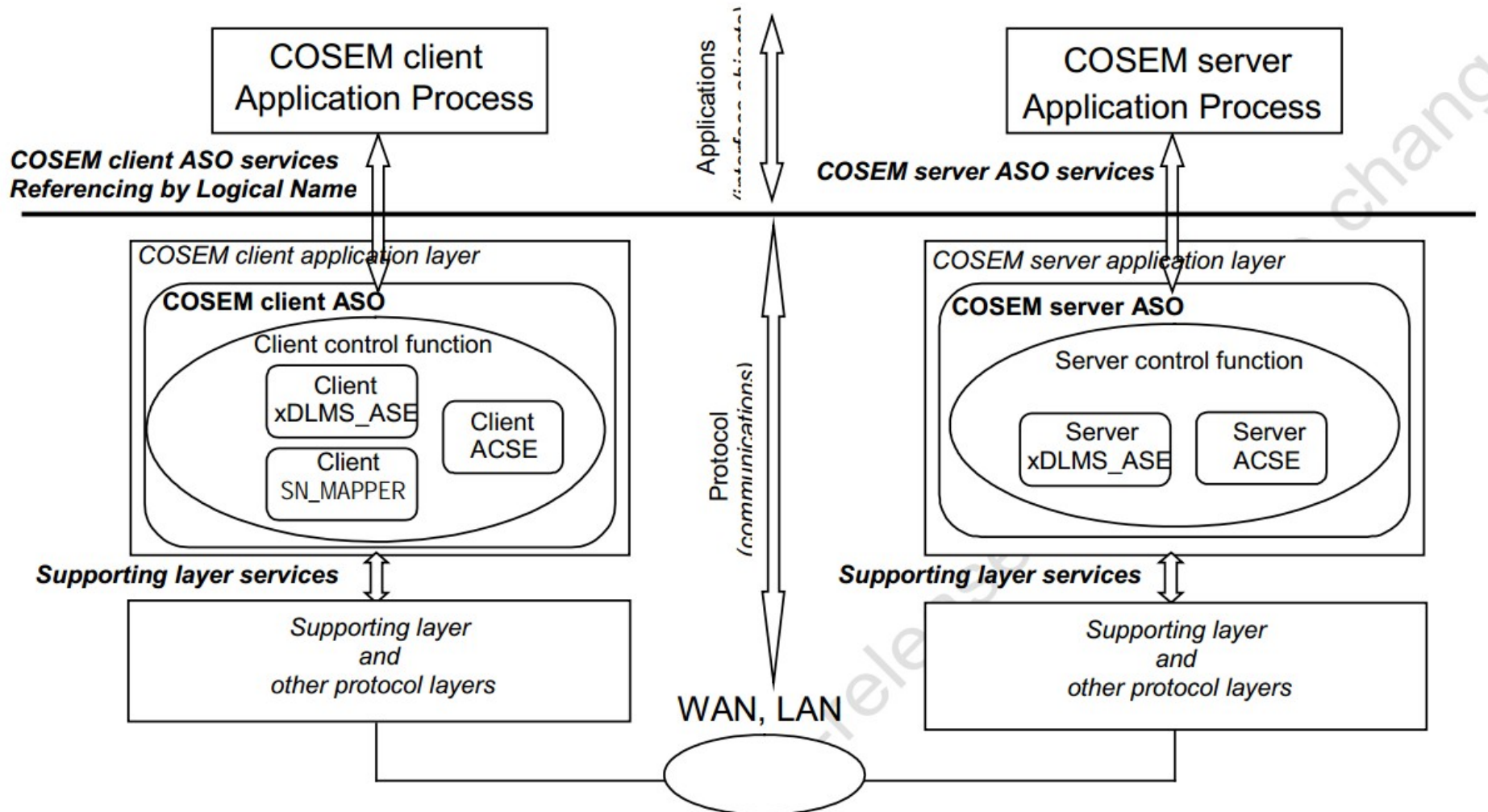
- A specific, application level service is available for this purpose: the protocol identification service.
- It is an optional application level service, allowing the client AE to obtain information after establishing a physical connection about the protocol stack implemented in the server.
- The protocol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

System Integration and Metering

- 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.

DLMS/COSEM Application Layer

Structure Of COSEM Application Layer



DLMS/COSEM Application Layer services

- ASO Services
- Services provided application association establishment and release
 - ➔ COSEM-OPEN
 - Confirmed
 - Unconfirmed
 - Pre-established
 - ➔ COSEM-RELEASE
 - ➔ COSEM-ABORT
- Services provided for the data transfer
 - ➔ xDLMS application service Element.
 - ➔ xDLMS Services types.
 - request/response type services
 - unsolicited services,
 - ➔ Referencing methods and service mapping.
 - LN referencing
 - SN referencing
 - ➔ Confirmed and unconfirmed services.

→ DLMS/COSEM request/response type services.

- GET
- SET
- ACTION
- ACCESS
- READ
- WRITE
- UNCONFIRMED WRITE

→ Unsolicited services.

→ Identifying Service invocations: the invoke_id parameter.

→ Priority service invocations : the priority parameter .

- High
- Normal

→ Selective access.

→ Multiple referencing .

→ Attribute_0 referencing.

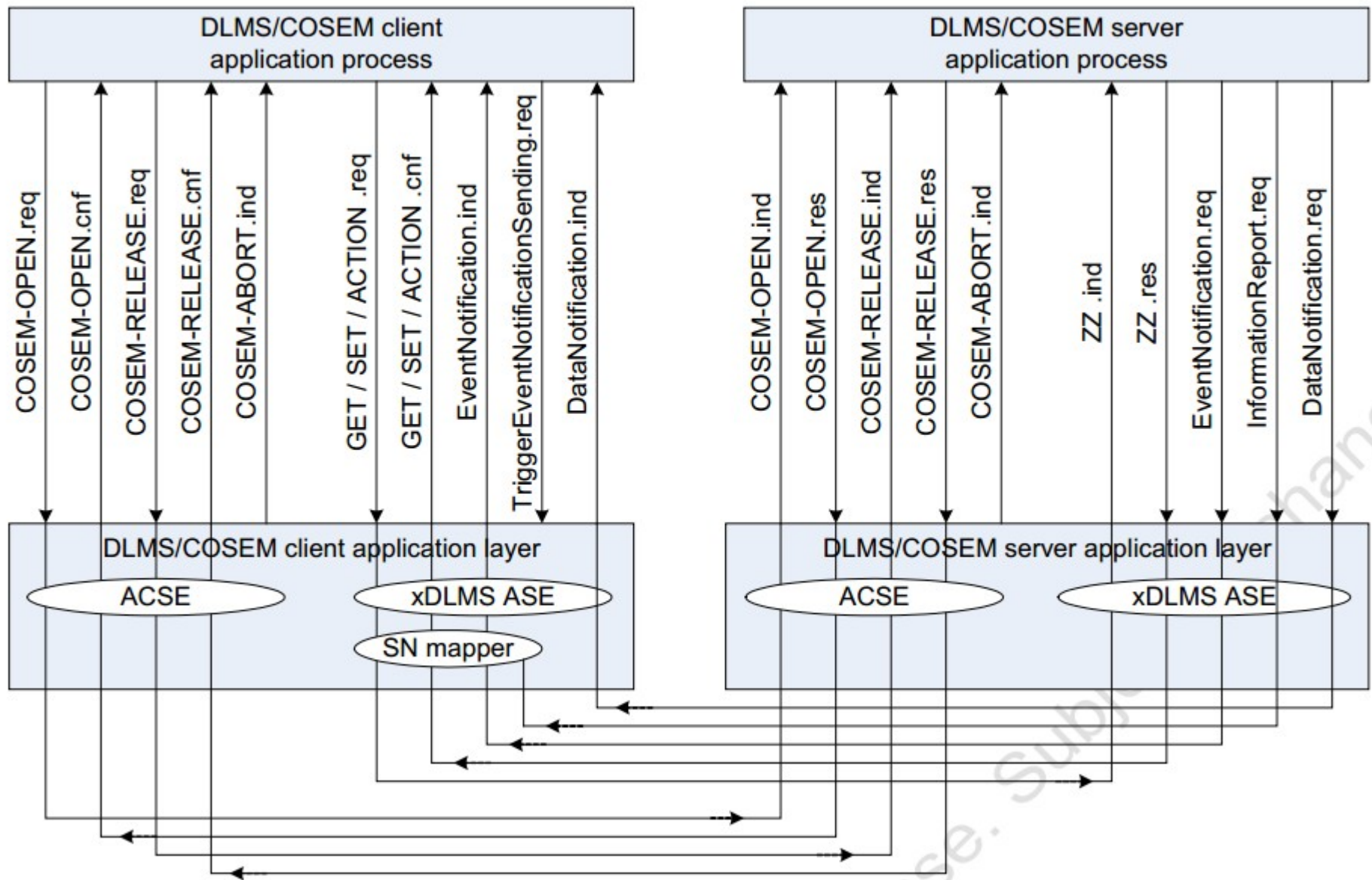
→ Transferring long service parameters.

- service specific block transfer mechanism.
- the general block transfer (GBT) mechanism

→ The general block transfer Mechanism.

- uni-directional block transfer
- bi-directional block transfer
- Streaming
- lost block recovery

➤ Layer management service .



COSEM application layer service specification

COSEM-OPEN service

- The function of the COSEM-OPEN service is to establish an AA between peer COSEM Application Entities (Aes).
- It uses the A-ASSOCIATE service of the ACSE.
- The COSEM-OPEN service provides only the framework for transporting this information.

Parameters

- The service parameters of the COSEM-OPEN.request service primitive, except the Protocol_Connection_Parameters, the User_Information parameter and – depending on the communication profile – the Service_Class parameter are carried by the fields of the AARQ APDU sent by the client.
- The service parameters of the COSEM-OPEN.response service primitive, except the Protocol_Connection_Parameters is carried by the fields of the AARE APDU sent by the server.

	.request	.indication	.response	.confirm
Protocol_Connection_Parameters	M	M (=)	M	M (=)
ACSE_Protocol_Version	U	U (=)	U	U (=)
Application_Context_Name	M	M (=)	M	M (=)
Called_AP_Title	U	U (=)	-	-
Called_AE_Qualifier	U	U(=)	-	-
Called_AP_Invocation_Identifier	U	U (=)	-	-
Called_AE_Invocation_Identifier	U	U (=)	-	-
Calling_AP_Title	C	C (=)	-	-
Calling_AE_Qualifier	U	U (=)	-	-
Calling_AP_Invocation_Identifier	U	U (=)	-	-
Calling_AE_Invocation_Identifier	U	U (=)	-	-
Local_Or_Remote	-	-	-	M
Result	-	-	M	M
Failure_Type	-	-	M	M
Responding_AP_Title	-	-	C	C (=)
Responding_AE_Qualifier	-	-	U	U (=)
Responding_AP_Invocation_Identifier	-	-	U	U (=)
Responding_AE_Invocation_Identifier	-	-	U	U (=)
ACSE_Requirements	U	U (=)	U	U (=)
Security_Mechanism_Name	C	C (=)	C	C (=)
Calling_Authentication_Value	C	C (=)	-	-
Responding_Authentication_Value	-	-	C	C (=)
Implementation_Information	U	U (=)	U	U (=)
Proposed_xDLMS_Context	M	M (=)	-	-
Dedicated_Key	C	C (=)	-	-
Response_Allowed	C	C (=)		
Proposed_DLMS_Version_Number	M	M (=)	-	-
Proposed_DLMS_Conformance	M	M (=)	-	-
Client_Max_Receive_PDU_Size	M	M (=)	-	-
Negotiated_xDLMS_Context	-	-	S	S (=)
Negotiated_DLMS_Version_Number	-	-	M	M (=)
Negotiated_DLMS_Conformance	-	-	M	M (=)
Server_Max_Receive_PDU_Size	-	-	M	M (=)
VAA_Name			M	M (=)
xDLMS_Initiate_Error			S	S (=)
User_Information	U	C (=)	-	-
Service_Class	M	M (=)	-	-

- **Protocol_Connection_Parameters parameter**
 - ➔ It is mandatory.
 - It contains all information necessary to use the layers of the communication profile, including the communication profile (protocol) identifier and the addresses required.
 - ➔ It identifies the participants of the AA.
 - ➔ The elements of this parameter are passed to the entities managing lower layer connections and to the lower layers as appropriate.
- **ACSE_Protocol_Version parameter**
 - ➔ It is optional.
 - ➔ If present, the default value shall be used.
- **The Application_Context_Name parameter**
 - ➔ It is mandatory.
 - ➔ In the request primitive, it holds the value proposed by the client.
 - ➔ In the response primitive, it holds the same value or the value supported by the server.
- The use of the **Called_AP_Title, Called_AE_Qualifier, Called_AP_Invocation_Identifier, Called_AE_Invocation_Identifier** parameters is optional.
- **Calling_AP_Title parameter**
 - ➔ It is conditional.
 - ➔ When the Application_Context_Name indicates an application context using ciphering, it shall carry the client system title .

- **Calling_AE_Qualifier parameter**
 - ➔ It is conditional.
 - ➔ When the Application_Context_Name indicates an application context using ciphering it may carry the public digital signature key certificate of the client.
- **Calling_AP_Invocation_Identifier**
 - ➔ It is optional.
- **Calling_AE_Invocation_Identifier parameter**
 - ➔ It is optional.
 - ➔ When present, it carries the identifier of the client-side user of the AA.
- **Local_or_Remote parameter**
 - ➔ It is mandatory.
 - ➔ It indicates the origin of the COSEM-OPEN.confirm primitive.
 - ➔ It is set to Remote if the primitive has been generated following the reception of an AARE APDU from the server.
 - ➔ It is set to Local if the primitive has been locally generated.
- **Result parameter**
 - ➔ It is mandatory.
 - ➔ In the case of remote confirmation, it indicates whether the Server accepted the proposed AA or not.
 - ➔ In the case of local confirmation, it indicates whether the client side protocol stack accepted the request or not.

- **Failure_Type parameter**
 - ➔ It is mandatory.
 - ➔ In the case of remote confirmation, it carries the information provided by the server.
 - ➔ In the case of local and negative confirmation, it indicates the reason for the failure.
- **Responding_AP_Title parameter**
 - ➔ It is conditional.
 - ➔ When the Application_Context_Name parameter indicates an application context using ciphering, it shall carry the server system title .
- **Responding_AE_Qualifier**
 - ➔ It is conditional.
 - ➔ When the Application_Context_Name indicates an application context using ciphering,
 - ➔ it may carry the public digital signature key certificate of the server.
- **Responding_AP_Invocation_Identifier and Responding AE_Invocation_Identifier Parameters**
 - ➔ It is optional.
- **ACSE_Requirements parameter**
 - ➔ It is optional.
 - ➔ It is used to select the optional authentication functional unit of the A-Associate service for the association

The presence of the **ACSE_requirements parameter** depends on the authentication mechanism used:

- in the case of Lowest Level security authentication, it shall not be present; only the Kernel functional unit is used;
- in the case of Low Level security (LLS) authentication it shall be present in the .request primitive and it may be present in the .response service primitive;
- in the case of High Level Security (HLS) authentication, it shall be present both in the .request and the .response service primitives.

➤ **Security_Mechanism_Name parameter**

- ➔ It is conditional.
- ➔ It is present only, if the authentication functional unit has been selected.
- ➔ If present, the .request primitive holds the value proposed by the client and the .response primitive holds the value required by the server, i.e. the one to be used by the client.

➤ **Calling_Authentication_Value parameter and the Responding_Authentication_Value parameters**

- ➔ They are conditional.
- ➔ They are present only, if the authentication functional unit has been selected. They hold the client authentication value / server authentication value respectively, appropriate for the Security_Mechanism_Name.

- **Implementation_Information parameter**
 - ➔ It is optional.
- **Proposed_xDLMS_Context parameter**
 - ➔ It holds the elements of the proposed xDLMS context.
 - ➔ It is carried by the xDLMS InitiateRequest APDU, placed in the user-information field of the AARQ APDU.
- **Dedicated_Key element**
 - ➔ It is conditional.
 - It may be present only, when the Application_Context_Name parameter indicates an application context using ciphering.
 - ➔ The dedicated key is used for dedicated ciphering of xDLMS APDUs exchanged within the AA established.
 - ➔ When the dedicated key is present, the xDLMS InitiateRequest APDU shall be authenticated and encrypted using the AES-GCM algorithm.
 - ➔ The xDLMS InitiateRequest APDU shall be ciphered the same way, when the dedicated key is not present, but it is necessary to protect the RLRQ APDU by including the ciphered xDLMS InitiateRequest in its user-information field.
- **Response_Allowed element**
 - ➔ It is conditional.
 - ➔ It indicates if the server is allowed to respond with an AARE APDU, i.e. if the AA to be established is confirmed (Response_Allowed == TRUE) or not confirmed (Response_Allowed == FALSE).

- **Proposed_DLMS_Version_Number** element holds the proposed DLMS version number.
- **Proposed_DLMS_Conformance** element holds the proposed conformance block.
- **Client_Max_Receive_PDU_Size** element holds the maximum length of the xDLMS APDUs the client can receive.
- If the xDLMS context proposed by the client is acceptable for the server, then the response service primitive shall contain the **Negotiated_xDLMS_Context parameter**. It is carried by the xDLMS InitiateResponse APDU, placed in the user-information field of the AARE APDU. If the xDLMS InitiateRequest APDU has been ciphered, the xDLMS InitiateResponse APDU shall be also ciphered the same way.
- **The Negotiated_DLMS_Version_Number** element holds the negotiated DLMS version number.
- **The Negotiated_DLMS_Conformance** element holds the negotiated conformance block.
- **The Server_Max_Receive_PDU_Size** element carries the maximum length of the xDLMS APDUs the server can receive.

- The **VAA_name** element carries the dummy value of 0x0007 in the case of LN referencing, and the **base_name** of the current Association object, 0xFA00, in the case of SN referencing.
- If the xDLMS context proposed by the client is not acceptable for the server, then the response service primitive shall carry the **xDLMS_Initiate_Error parameter**.
- It is carried by the ConfirmedServiceError APDU, with appropriate diagnostic elements, placed in the user-information field of the AARE APDU.
- The **User_Information parameter** is optional. If present, it shall be passed on to the supporting layer, provided it is capable to carry it. The .indication primitive shall then contain the user-specific information carried by the supporting lower protocol layer(s).
- The **Service_Class parameter** is mandatory. It indicates whether the service is invoked in a confirmed or in an unconfirmed manner. The handling of this parameter may depend on the communication profile.


The COSEM-RELEASE service

- The function of the COSEM-RELEASE service is to gracefully release an existing AA. Depending on the way it is invoked, it uses the A-RELEASE service of the ACSE or not.
- The **Use_RLRQ_RLRE parameter** in the .request primitive is optional.
- If present, its value may be FALSE (default) or TRUE.
- It indicates whether the ACSE A-RELEASE service – involving an RLRQ / RLRE APDU exchange – should be used or not.
- The Use_RLRQ_RLRE parameter in the .response primitive is conditional. If it was present in the .indication primitive and its value was TRUE, it shall also be present and its value shall be TRUE. Otherwise, it shall not be present or its value shall be FALSE.
- If the value of the Use_RLRQ_RLRE parameter is FALSE, then the AA can be released by disconnecting the supporting layer of the AL.
- The **Reason parameter** is optional. It may be present only if the value of the Use_RLRQ_RLRE is TRUE. It is carried by the reason field of the RLRQ / RLRE APDU respectively.

	.request	.indication	.response	.confirm
Use_RLRQ_RLRE	U	C(=)	C(=)	-
Reason	U	U (=)	U	U (=)
Proposed_xDLMS_Context	C	C (=)	–	–
Negotiated_xDLMS_Context	–	–	C	C (=)
Local_Or_Remote	–	–	–	M
Result	–	–	M	M
Failure_Type	–	–	–	C
User_Information	U	C (=)	U	C (=)

- When used on the .request primitive, this parameter identifies the general level of urgency of the request.
- When used on the .response primitive, this parameter identifies information about why the acceptor accepted or rejected the release request.
- Note, that in DLMS/COSEM the server cannot reject the release request.
- The **Proposed_xDLMS_Context parameter** is conditional. It is present only if the value of the Use_RLRQ_RLRE is TRUE and the AA to be released has been established with an application context using ciphering. This option allows securing the COSEM-RELEASE service, and avoiding thereby a denial-of-service attack that may be carried out by unauthorized releasing of the AA.
- In the .request primitive, the Proposed_xDLMS_Context parameter shall be the same as in the COSEM-OPEN.request service primitive, having established the AA to be released. It is carried by the xDLMS InitiateRequest APDU, authenticated and encrypted the same way as in the AARQ and placed in the user-information field of the RLRQ APDU.

- If the xDLMS InitiateRequest APDU can be successfully deciphered, then the .response primitive shall carry the same **Negotiated_xDLMS_Context parameter** as in the COSEM-OPEN.response primitive. It is carried by the xDLMS InitiateResponse APDU, authenticated and encrypted the same way as in the AARE and placed in the user-information field of the RLRE APDU.
- Otherwise, the RLRQ APDU is silently discarded.
- The **Local_or_Remote parameter** is mandatory. It indicates the origin of the COSEM-RELEASE.confirm primitive.
 - It is set to Remote if either:
 - ➔ a RLRE APDU has been received from the server; or
 - ➔ a disconnect confirmation service primitive has been received.
 - It is set to Local if the primitive has been locally generated.
- The **Result parameter** is mandatory. In the .response primitive, it indicates whether the server AP can accept the request to release the AA or not. As servers cannot refuse such requests, its value should normally be SUCCESS unless the AA referenced does not exist.



The **Failure_Type parameter** is conditional. It is present if Result == ERROR. In this case, it indicates the reason for the failure. It is a locally generated parameter on the client side.

The **User_Information parameter** in the .request primitive is optional. If present, it is passed to the supporting layer, provided it is able to carry it. The .indication primitive contains then the user- specific information carried by the supporting lower protocol layer(s). Similarly, it is optional in the .response primitive. If present, it is passed to the supporting layer. In the .confirm primitive, it may be present only when the service is remotely confirmed. It contains then the user-specific information carried by the supporting lower protocol layer(s).

COSEM-ABORT service

- The function of the COSEM-ABORT service is to indicate an unsolicited disconnection of the supporting layer.
- The **Diagnostics parameter** is optional. It shall indicate the possible reason for the disconnection, and may carry lower protocol layer dependent information as well.

		.indication
Diagnostics		U

Protection and general block transfer parameters

	.request	.indication	.response	.confirm
Additional_Service_Parameters	U	–	U (=)	
Invocation_Type	M	–	M (=)	–
Security_Options	C	–	C (=)	–
General_Block_Transfer_Parameters	C	–	C (=)	–
Block_Transfer_Streaming	M	–	M (=)	–
Block_Transfer_Window	M	–	M (=)	–
Service_Parameters	M	–	M (=)	–
Additional_Service_Parameters	–	U	–	U (=)
Invocation_Type	–	U	–	U (=)
Security_Status	–	C	–	C (=)
General_Block_Transfer__Parameters	–	C	–	C (=)
Block_Transfer_Window	–	M	–	M (=)
Service_Parameters	–	M	–	M (=)
Protection_Element	–	C	–	C (=)
NOTE The service primitives available depend on the kind of the service.				

- The **Additional_Service_Parameters** are present only if ciphering or GBT is used.
- The **Invocation_Type parameter** is mandatory: it indicates if the service invocation is complete or partial. Possible values: COMPLETE, FIRST-PART, ONE-PART and LAST-PART.
- The **Security_Options parameter** is conditional: it is present only if the application context is a ciphered one, the .request / .response service primitive has to be ciphered and Invocation_Type = COMPLETE or FIRST-PART. It determines the protection to be applied by the AL.
- The **General_Block_Transfer_Parameters** parameter is conditional: it is present only if general block transfer (GBT) is used and Invocation_Type = COMPLETE or FIRST-PART. It provides information on the GBT streaming capabilities:
 - ➔ the **Block_Transfer_Streaming parameter** is present only in .request and .response service primitives. It is passed by the AP to the AL to indicate if the AL is allowed to send General-Block- Transfer APDUs using streaming (TRUE) or not (FALSE);
 - ➔ the **Block_Transfer_Window parameter** indicates the window size supported, i.e. the maximum number of blocks that can be received in a window.
 - ➔ The streaming process itself is managed by the AL.

- The **Service_Parameters** are mandatory: they include the parameters of xDLMS service invocations. If `Invocation_Type != COMPLETE`, then it includes a part of the service parameters.
- The **Security_Status parameter** is conditional: it is present only if cryptographic protection has been applied. It carries information on the protection that has been verified / removed by the AL. It may be present in all type of service invocations.
- The **Protection_Element parameter** is conditional: it is present only if the APDU has been authenticated or signed.

Security_Status	–	C	–	C (=)
Security_Status_Element {Security_Status_Element}	–	M	–	M (=)
Security_Protection_Type		M		M (=)
Glo_Ciphering	–	S	–	S (=)
Ded_Ciphering	–	S	–	S (=)
General_Glo_Ciphering	–	S	–	S (=)
General_Ded_Ciphering	–	S	–	S (=)
General_Ciphering	–	S	–	S (=)
General_Signing	–	S	–	S (=)
<i>With General_Glo_Ciphering and General_Ded_Ciphering</i>				
System_Title	–	U	–	U (=)
<i>With General_Ciphering and General_Signing</i>				
Transaction_Id	–	U	–	U (=)
Originator_System_Title	–	U	–	U (=)
Recipient_System_Title	–	U	–	U (=)
Date_Time	–	U	–	U (=)
Other_Information	–	U	–	U (=)
<i>With General_Ciphering</i>				
Key_Info_Status	–	C	–	C (=)
Identified_Key_Status	–	S	–	S (=)
Wrapped_Key_Status	–	S	–	S (=)
Agreed_Key_Status	–	S	–	S (=)
<i>With Glo_Ciphering, Ded_Ciphering, General_Ded_Ciphering, General_Glo_Ciphering, General_Ciphering</i>				
Security_Control	–	M	–	M (=)
<i>The protection element is present when authentication or digital signature is applied.</i>				
Protection_Element {Protection_Element}	–	C	–	C (=)
Invocation_Counter	–	C	–	C (=)
Authentication_Tag	–	C	–	C (=)
Signature	–	C	–	C (=)

- the **Key_Info_Options subparameter** is mandatory: it carries information on the symmetric key that has been used by the originator / is to be used by the recipient. The key information is sent received as part of the ciphered APDU:
 - ➔ **Identified_Key_Options** : it can be used when the partners share the key; this may be the global unicast encryption key or the global broadcast encryption key;
 - ➔ **Wrapped_Key_Options** : in this case, a wrapped key is sent;
 - ➔ **Agreed_Key_Options** : in this case, the partners use a Diffie-Hellman key agreement scheme to agree on the key;

NOTE Key wrapping and key agreement cannot be used for broadcast.

- **Security_Control**: contains the Security Control byte.
- **The Protection_Element parameter** is conditional: it shall be present if the APDU has been authenticated or digitally signed. It may be present in all type of service invocations, but it may be empty if it is not yet available (this may occur in the case when general block transfer is used). It contains:
 - ➔ in the case of General-Ciphering, the Invocation_Counter, holding the invocation field of the initialization vector,
 - ➔ in the case when the APDU has been authenticated, the authentication tag;
 - ➔ in the case when the APDU has been signed, the digital signature.

Secure Control byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3..0
Compression	Key_Set	E	A	Security_Suite_Id
The Key_Set bit is not relevant and shall be set to 0 when the service specific dedicated ciphering, the general-dedicated-ciphering or the general-ciphering APDUs are used.				

GET service

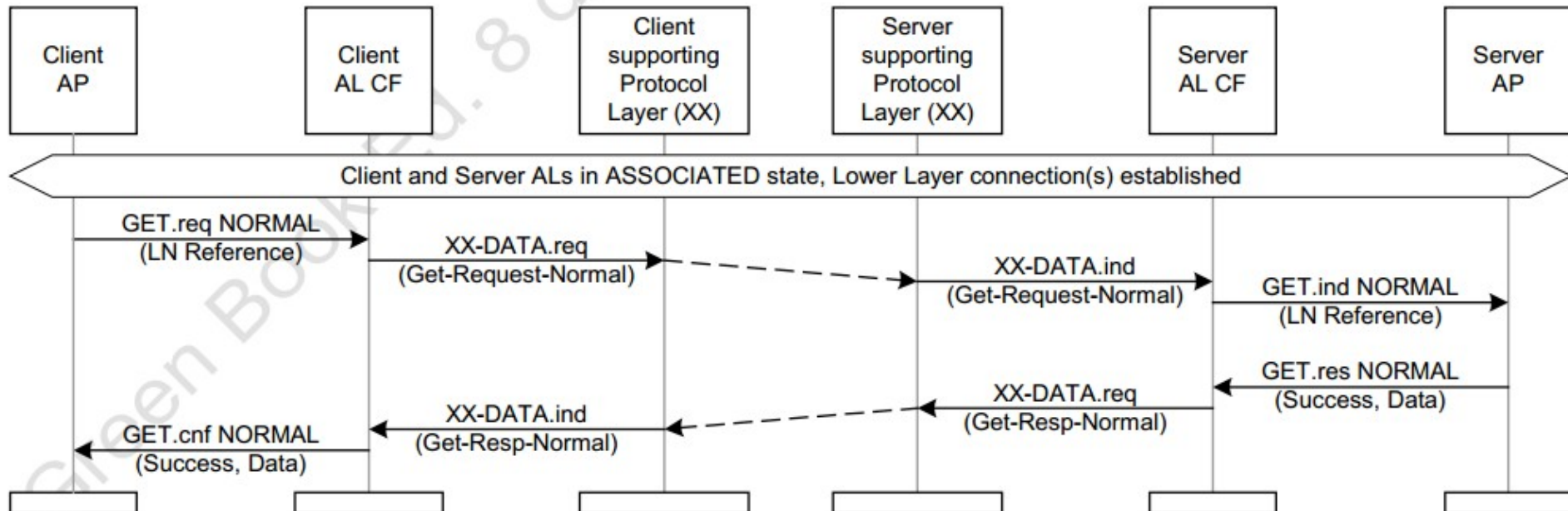
- The GET service is used with LN referencing.
- It can be invoked in a confirmed or unconfirmed manner.
- Its function is to read the value of one or more COSEM interface object attributes.
- The result can be delivered in a single response or – if it is too long to fit in a single response – in multiple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

	.request	.indication	.response	.confirm
Invoke_Id	M	M (=)	M (=)	M (=)
Priority	M	M (=)	M (=)	M (=)
Service_Class	M	M (=)	–	–
Request_Type	M	M (=)	–	–
COSEM_Attribute_Descriptor { COSEM_Attribute_Descriptor }	C	C (=)	–	–
COSEM_Class_Id	M	M (=)		
COSEM_Object_Instance_Id	M	M (=)		
COSEM_Object_Attribute_Id	M	M (=)		
Access_Selection_Parameters	U	U (=)		
Access_Selector	M	M (=)		
Access_Parameters	M	M (=)		
Block_Number	C	C (=)	–	–
Response_Type	–	–	M	M (=)
Result	–	–	M	M (=)
Get_Data_Result { Get_Data_Result }	–	–	S	S (=)
Data			S	S (=)
Data_Access_Result			S	S (=)
DataBlock_G	–	–	S	S (=)
Last_Block			M	M (=)
Block_Number			M	M (=)
Result			M	M (=)
Raw_Data			S	S (=)
Data_Access_Result			S	S (=)

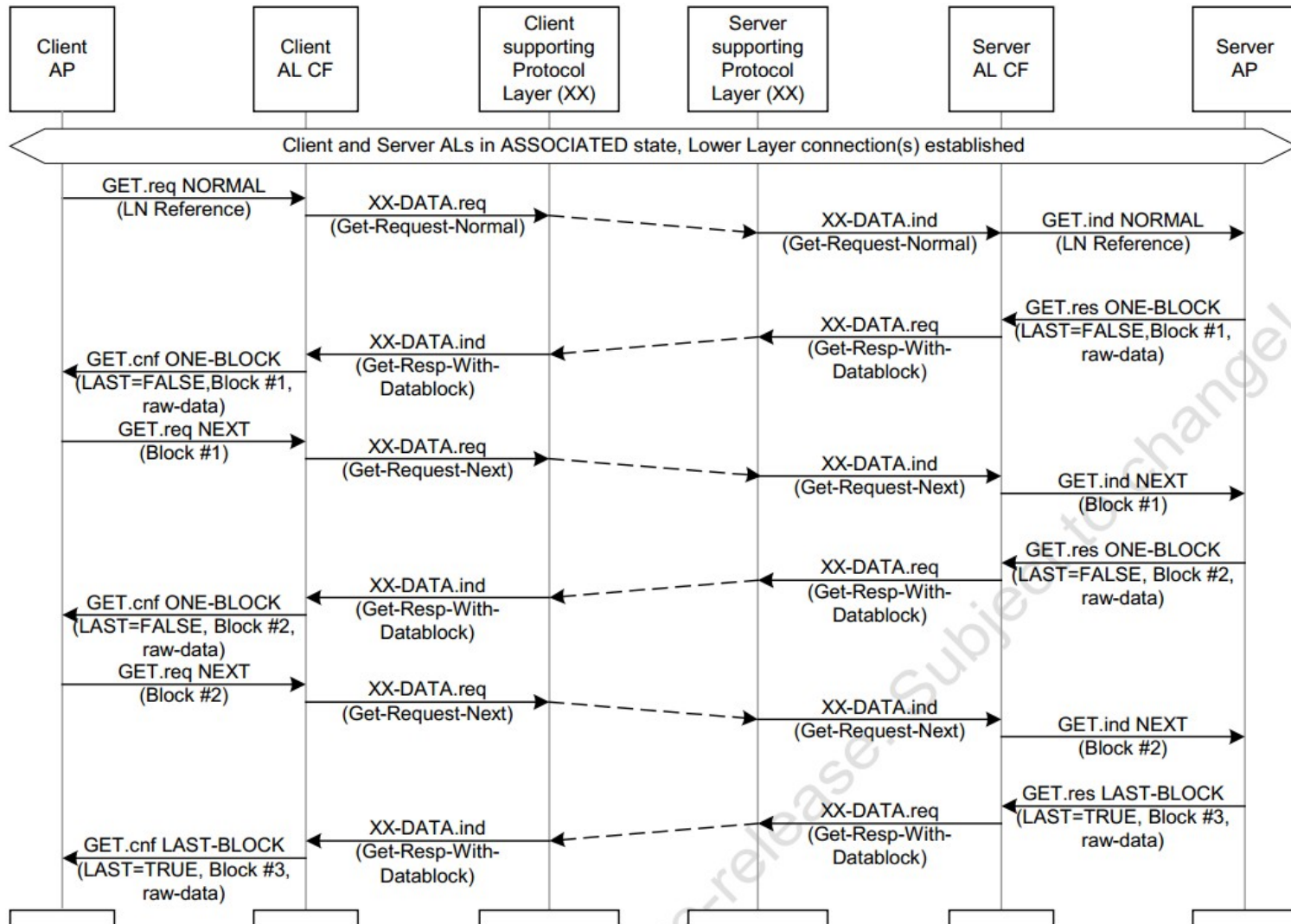
NOTE For security parameters, see Table 50.

Request type		Response type	
NORMAL	The value of a single attribute is requested.	NORMAL	The complete result is delivered.
		ONE-BLOCK	One block of the result is delivered.
NEXT	The next data block is requested.	ONE-BLOCK	As above.
		LAST-BLOCK	The last block of the result is delivered.
WITH-LIST	The value of a list of attributes is requested.	WITH-LIST	The complete result is delivered.
		ONE-BLOCK	As above.
NOTE The same Response_Type may be present more than once, to show the possible responses to each kind of request.			

GET service without block transfer



GET service with block transfer



SET service

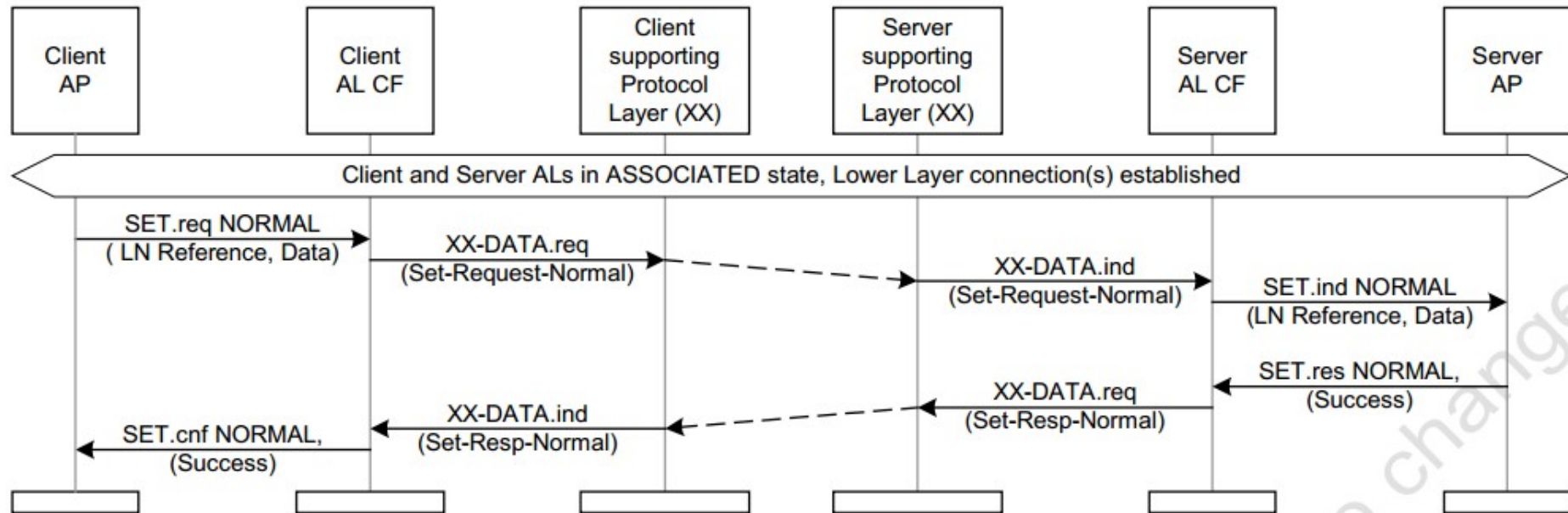
- The SET service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner.
- Its function is to write the value of one or more COSEM interface object attributes.
- The data to be written can be sent in a single request or – if it is too long to fit in a single request – in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

	.request	.indication	.response	.confirm
Invoke_Id	M	M (=)	M (=)	M (=)
Priority	M	M (=)	M (=)	M (=)
Service_Class	M	M (=)	–	–
Request_Type	M	M (=)	–	–
COSEM_Attribute_Descriptor { COSEM_Attribute_Descriptor }	C	C (=)	–	–
COSEM_Class_Id	M	M (=)		
COSEM_Object_Instance_Id	M	M (=)		
COSEM_Object_Attribute_Id	M	M (=)		
Access_Selection_Parameters	U	U (=)		
Access_Selector	M	M (=)		
Access_Parameters	M	M (=)		
Data {Data }	C	C (=)	–	–
DataBlock_SA	C	C (=)	–	–
Last_Block	M	M (=)		
Block_Number	M	M (=)		
Raw_Data	M	M (=)		
Response_Type	–	–	M	M (=)
Result { Result }	–	–	C	C (=)
Block_Number	–	–	C	C (=)
NOTE For security parameters, see Table 50.				

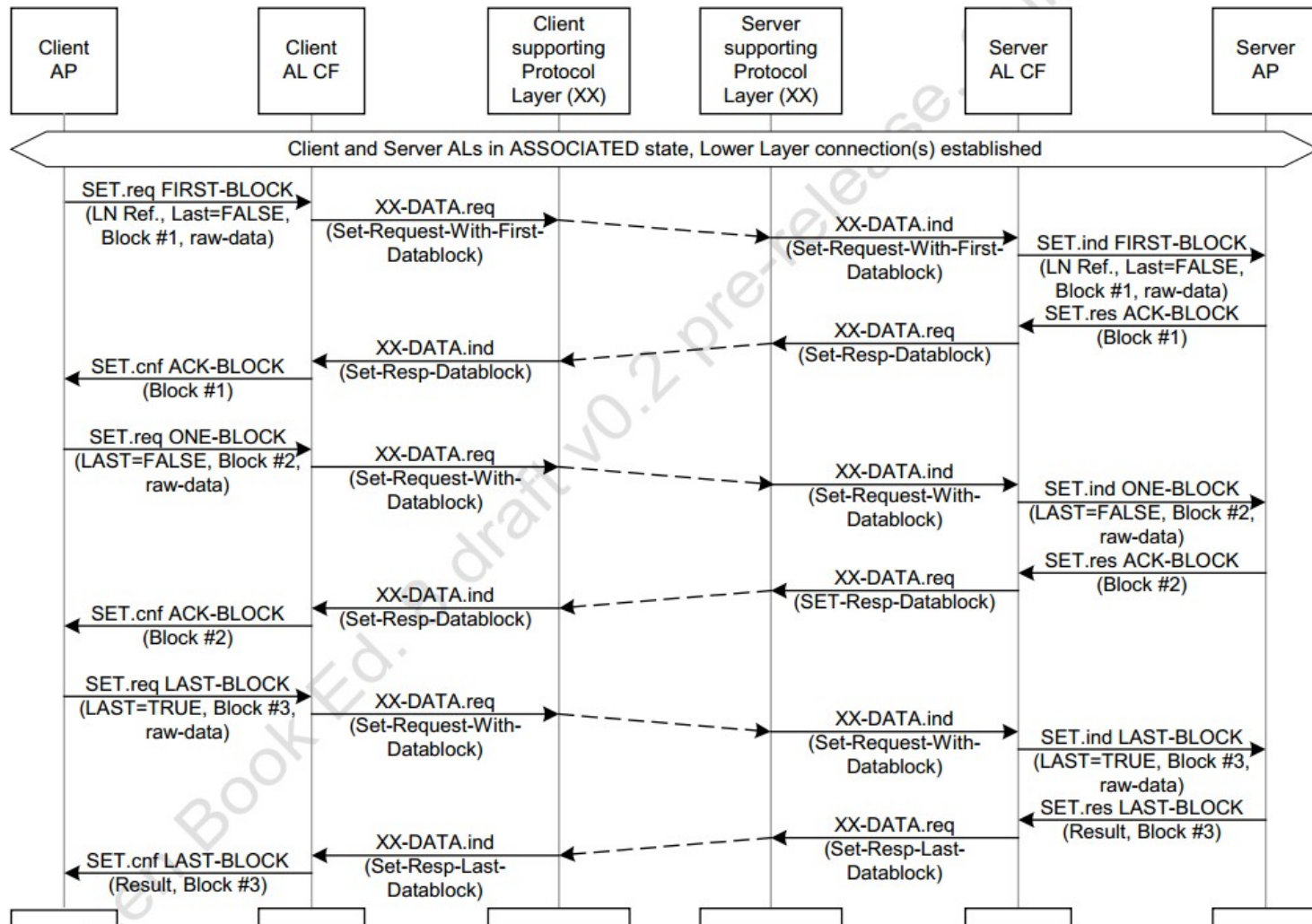
Request type		Response type	
NORMAL	The reference of a single attribute and the complete data to be written is sent.	NORMAL	The result is delivered.
FIRST-BLOCK	The reference of a single attribute and the first block of the data to be written is sent.	ACK-BLOCK	The correct reception of the block is acknowledged.
ONE-BLOCK	One block of the data to be written is sent.		
LAST-BLOCK	The last block of the data to be written is sent.	LAST-BLOCK	The correct reception of the last block is acknowledged and the result is delivered.
		LAST-BLOCK-WITH-LIST	The correct reception of the last block is acknowledged and the list of results is delivered.
WITH-LIST	The reference of a list of attributes and the complete data to be written is sent.	WITH-LIST	The list of results is delivered.
FIRST-BLOCK-WITH-LIST	The reference of a list of attributes and the first block of data to be written is sent.	ACK-BLOCK	The correct reception of the block is acknowledged.

NOTE The same Response_Type may be present more than once, to show the possible responses to each request.

SET service without block transfer



SET service with block transfer



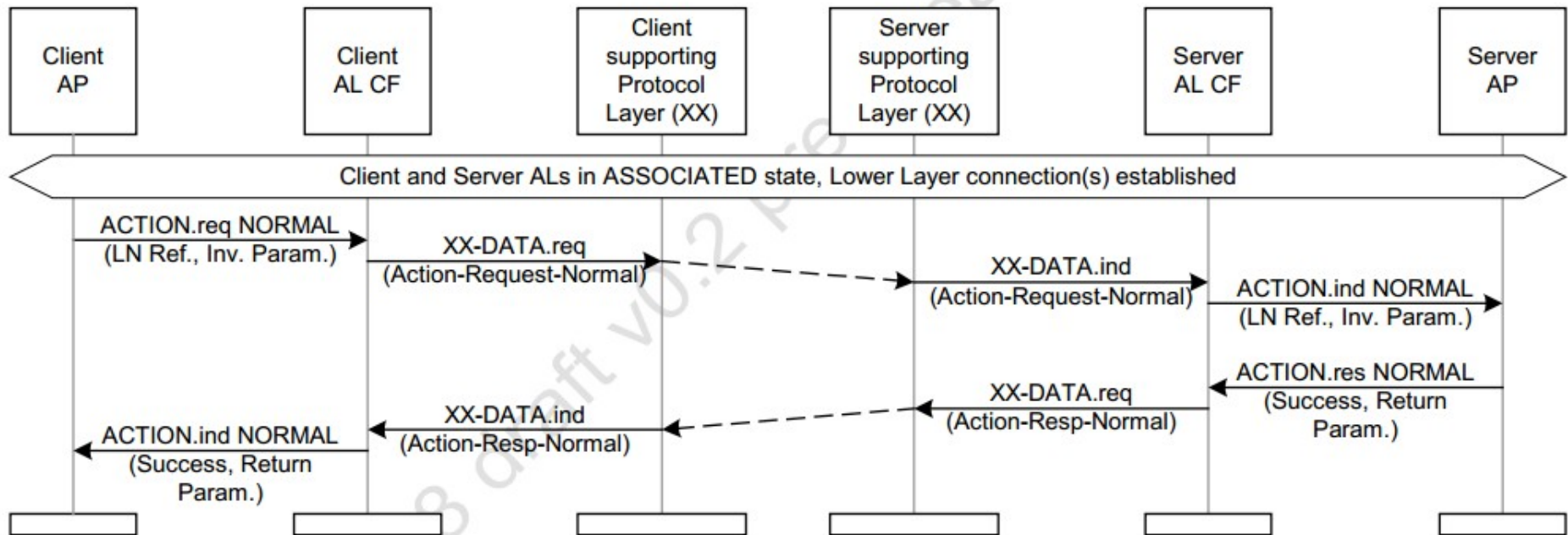
ACTION service

- The ACTION service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner.
- Its function is to invoke one or more COSEM interface objects methods. It comprises two phases:
 - ➔ in the first phase, the client sends the reference(s) of the method(s) to be invoked, with the method invocation parameters necessary;
 - ➔ in the second phase, after invoking the methods, the server sends back the result and the return parameters generated by the invocation of the method(s), if any.
- If the method invocation parameters are too long to fit in a single request, they are sent in multiple requests (block transfer from the client to the server). If the result and the return parameters are too long to fit in a single response, they are returned in multiple responses (block transfer from the server to the client.)

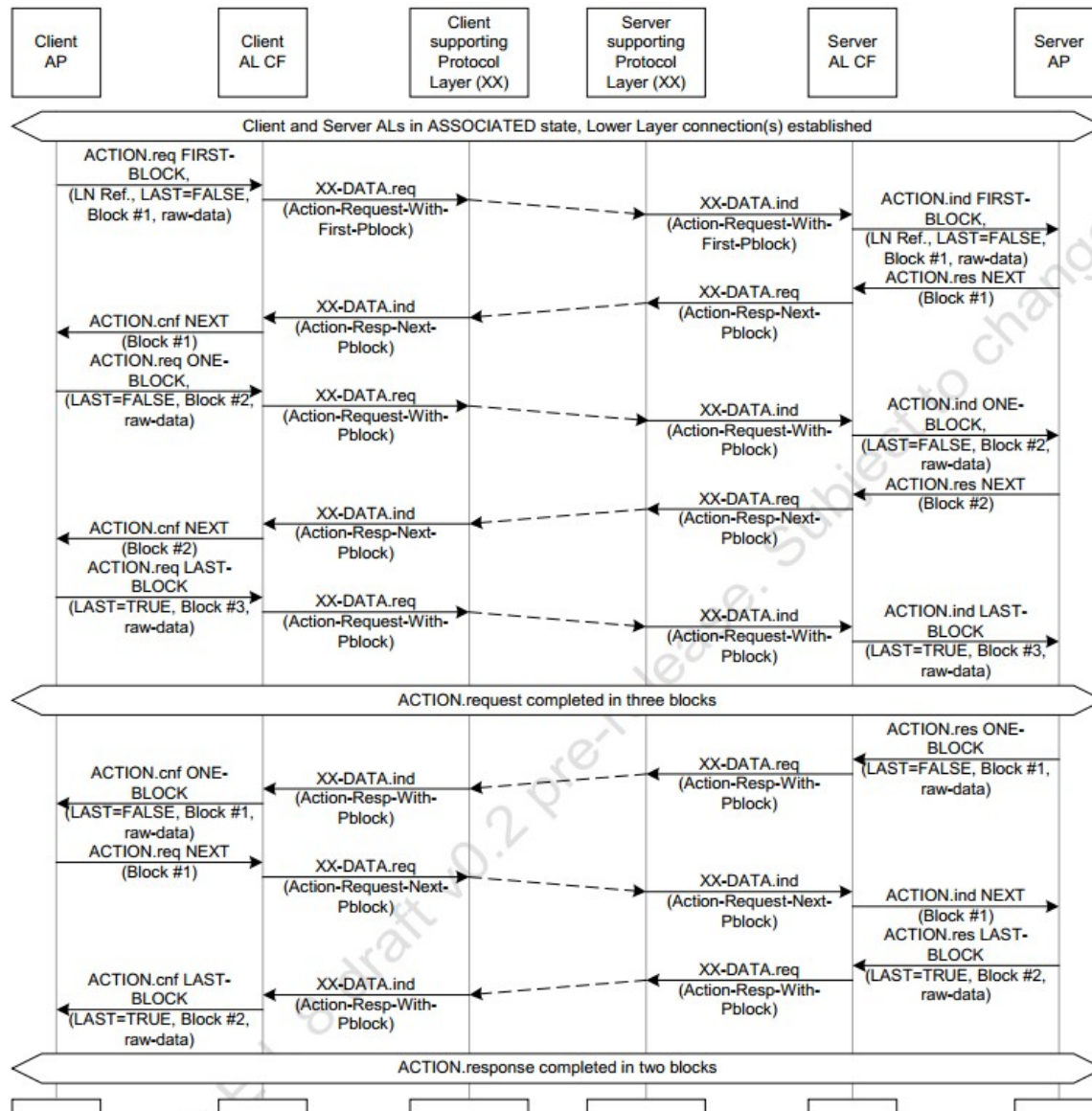
	.request	.indication	.response	.confirm
Invoke_Id	M	M (=)	M (=)	M (=)
Priority	M	M (=)	M (=)	M (=)
Service_Class	M	M (=)	–	–
Request_Type	M	M (=)	–	–
COSEM_Method_Descriptor { COSEM_Method_Descriptor }	C	C (=)	–	–
COSEM_Class_Id	M	M (=)		
COSEM_Object_Instance_Id	M	M (=)		
COSEM_Object_Method_Id	M	M (=)		
Method_Invocation_Parameters { Method_Invocation_Parameters }	U	U (=)	–	–
Response_Type	-	-	M	M (=)
Action_Response { Action_Response }	–	–	M	M (=)
Result			M	M (=)
Response_Parameters			U	U (=)
Data			S	S (=)
Data_Access_Result			S	S (=)
DataBlock_SA	C	C (=)	C	C (=)
Last_Block	M	M (=)	M	M (=)
Block_Number	M	M (=)	M	M (=)
Raw_Data	M	M (=)	M	M (=)
Block_Number	C	C (=)	C	C (=)
NOTE For security parameters, see Table 50.				

Request type		Response type	
NORMAL	The reference of a single method and the complete method invocation parameter is sent.	NORMAL	The result and the complete return parameter are sent.
		ONE-BLOCK	One block of the result and of the return parameter is sent.
NEXT	The next data block is requested.	ONE-BLOCK	As above.
		LAST-BLOCK	The last block of the result(s) and of the return parameter(s) is sent.
FIRST-BLOCK	The reference of a single method and the first block of the method invocation parameters is sent.	NEXT	The correct reception of the block is acknowledged.
ONE-BLOCK	One block of the method invocation parameters is sent.		
LAST-BLOCK	The last block of the method invocation parameters is sent.	NORMAL	As above.
		ONE-BLOCK	As above.
WITH-LIST	The reference of a list of methods and the complete list of method invocation parameters is sent.	WITH-LIST	The complete list of results and return parameters is sent.
		ONE-BLOCK	See above.
WITH-LIST-AND-FIRST-BLOCK	The reference of a list of methods and the first block of the method invocation parameters is sent.	NEXT	The correct reception of the block is acknowledged.

ACTION service without block



ACTION service with block



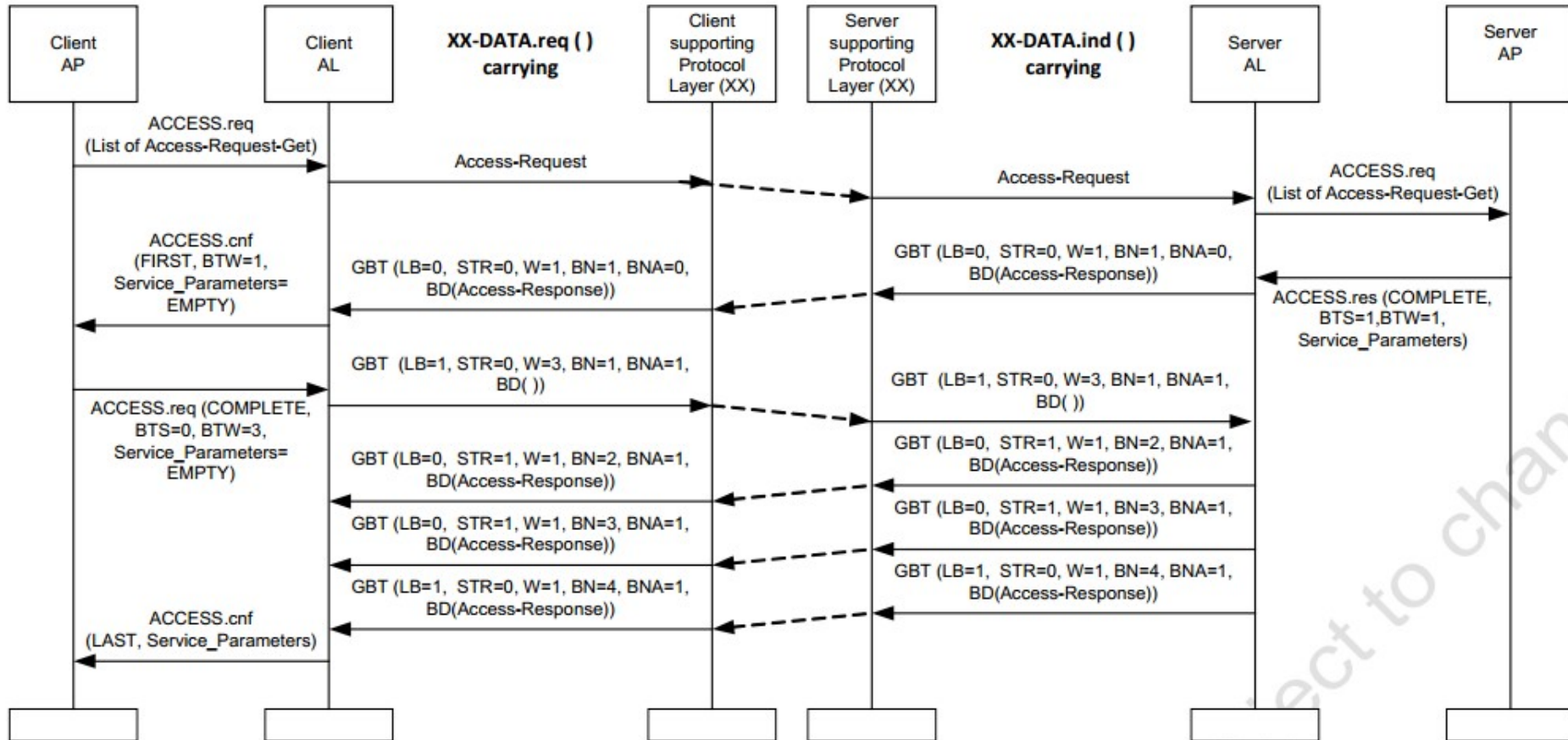
ACCESS service

- Unified WITH-LIST service to improve efficiency
- Specific variants for selective access
- Long_Invoke_Id parameter
- Self-descriptive responses
- Failure management
- Time stamp as a service parameter
- Separation of messaging, block transfer and cryptographic message protection
- Presence of data in service primitives

	.request	.indication	.response	.confirm
Long_Invoke_Id	M	M (=)	M (=)	M (=)
Self_Descriptive	M	M (=)	M (=)	M (=)
Processing_Option	M	M (=)	M (=)	M (=)
Service_Class	M	M (=)	M (=)	M (=)
Priority	M	M (=)	M (=)	M (=)
Date_Time	U	U (=)	U	U (=)
Access_Request_Body	M	M (=)	-	-
Access_Request_Specification	M	M (=)	-	-
{ Access_Request_Specification }				
Access_Request_Get	U	U (=)	-	-
Cosem_Attribute_Descriptor	M	M (=)	-	-
Access_Request_Set	U	U (=)	-	-
Cosem_Attribute_Descriptor	M	M (=)	-	-
Access_Request_Action	U	U (=)	-	-
Cosem_Method_Descriptor	M	M (=)	-	-
Access_Request_Get_With_Selection	U	U (=)	-	-
Cosem_Attribute_Descriptor	M	M (=)	-	-
Access_Selection	M	M (=)	-	-
Access_Selector	M	M (=)	-	-
Access_Parameters	M	M (=)	-	-
Access_Request_Set_With_Selection	U	U (=)	-	-
Cosem_Attribute_Descriptor	M	M (=)	-	-
Access_Selection	M	M (=)	-	-
Access_Selector	M	M (=)	-	-
Access_Parameters	M	M (=)	-	-
Access_Request_List_Of_Data	M	M (=)	-	-
Data { Data }			-	-
Access_Response_Body	-	-	M	M (=)
Access_Request_Specification	-	-	C (=) ¹	C (=)
{ Access_Request_Specification }				
Access_Response_List_Of_Data	-	-	M	M (=)
Data { Data }				
Access_Response_Specification	-	-	M	M (=)
{ Access_Response_Specification }				
Access_Response_Get	-	-	C	C (=)
Result	-	-	M	M (=)
Access_Response_Set	-	-	C	C (=)
Result	-	-	M	M (=)
Access_Response_Action	-	-	C	C (=)
Result	-	-	M	M (=)

¹ When the Access_Request_Specification service parameter is present in Access_Response_Body, then its value shall be the same as in the .request / .indication primitive.

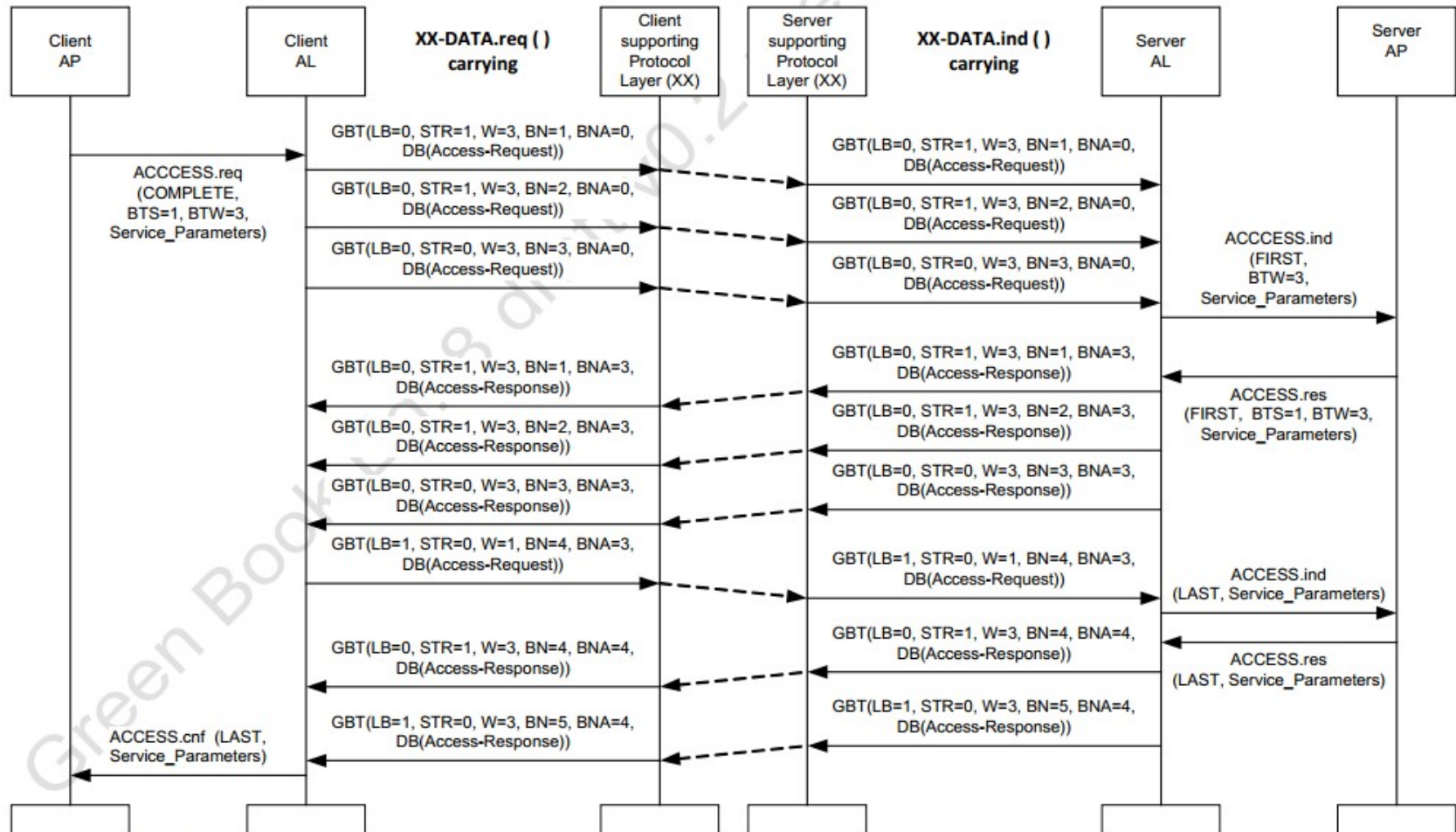
Access Service with long response



BTS: Block_Transfer_Streaming, BTW: Block_Transfer_Window,

GBT: General-Block-Transfer APDU, LB: Last block, STR: Streaming, W: Window, BN: Block number, BNA: Block number acknowledged, BD: Block data

Access Service with long response And Long request



BTS: Block_Transfer_Streaming, BTW: Block_Transfer_Window,

GBT: General-Block-Transfer APDU, LB: Last block, STR: Streaming, W: Window, BN: Block number, BNA: Block number acknowledged, BD: Block data

Data Notification service

- The DataNotification service is an unsolicited, unconfirmed service.
- It is used by the server to push data to the client.
- It is an unconfirmed service.

	.request	.indication
Long_Invoke_Id	M	M (=)
Self_Descriptive	–	–
Processing_Option	–	–
Service_Class	–	–
Priority	M	M (=)
Date_Time	U	U (=)
Notification_Body	M	M (=)

EventNotification service

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

	.request	.indication
Time	U	U (=)
Application_Addresses	U	U (=)
COSEM_Attribute_Descriptor	M	M (=)
COSEM_Class_Id	M	M (=)
COSEM_Object_Instance_Id	M	M (=)
COSEM_Object_Attribute_Id	M	M (=)
Attribute_Value	M	M (=)

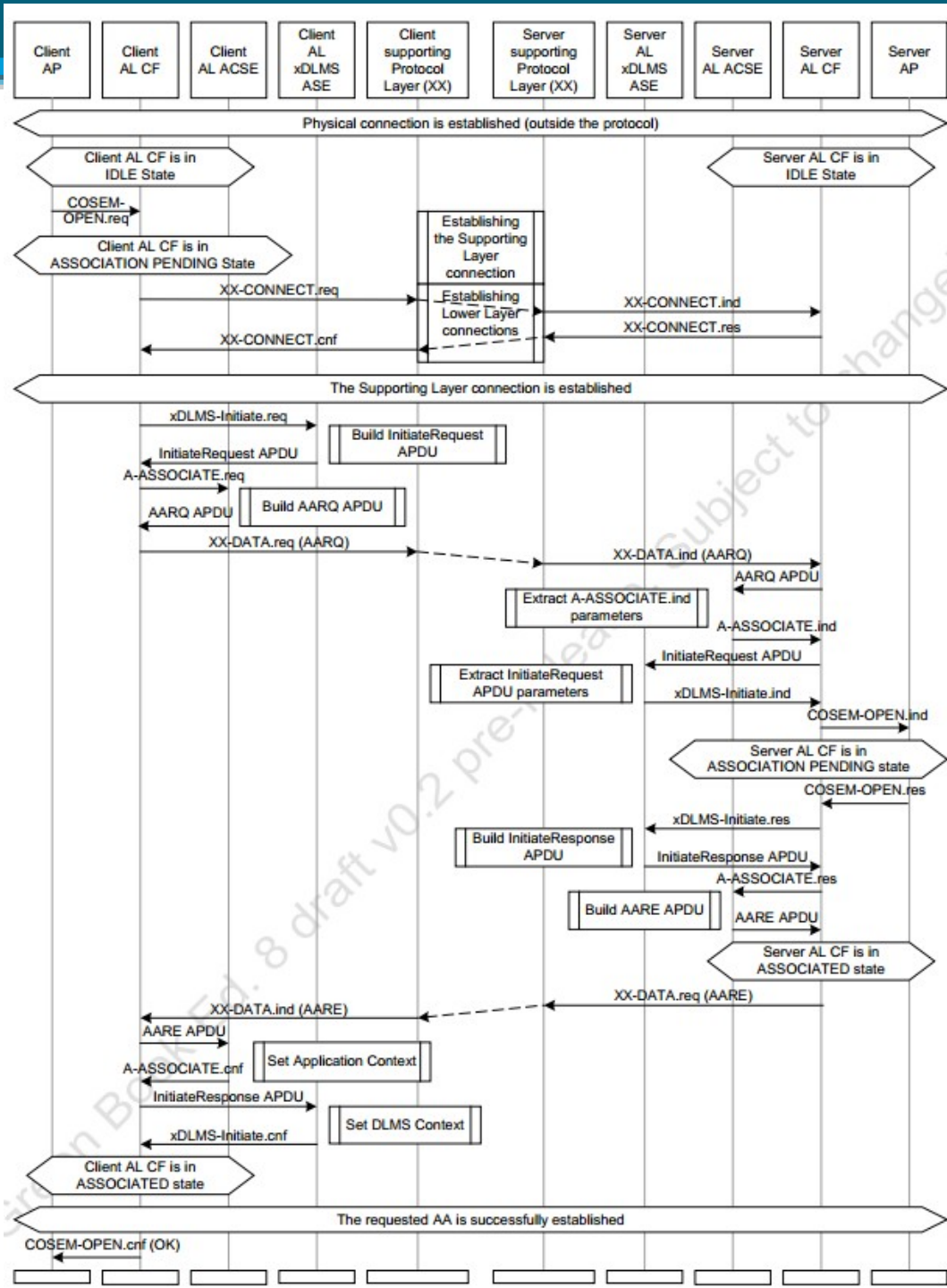
TriggerEventNotificationSending service

- The function of the TriggerEventNotificationSending service is to trigger the server by the client to send the frame carrying the EventNotification.request APDU.
- This service is necessary in the case of protocols, when the server is not able to send a real non- solicited EventNotification.request APDU.

	.request
Protocol_Parameters	M

Protocol For Application Associations Establishment

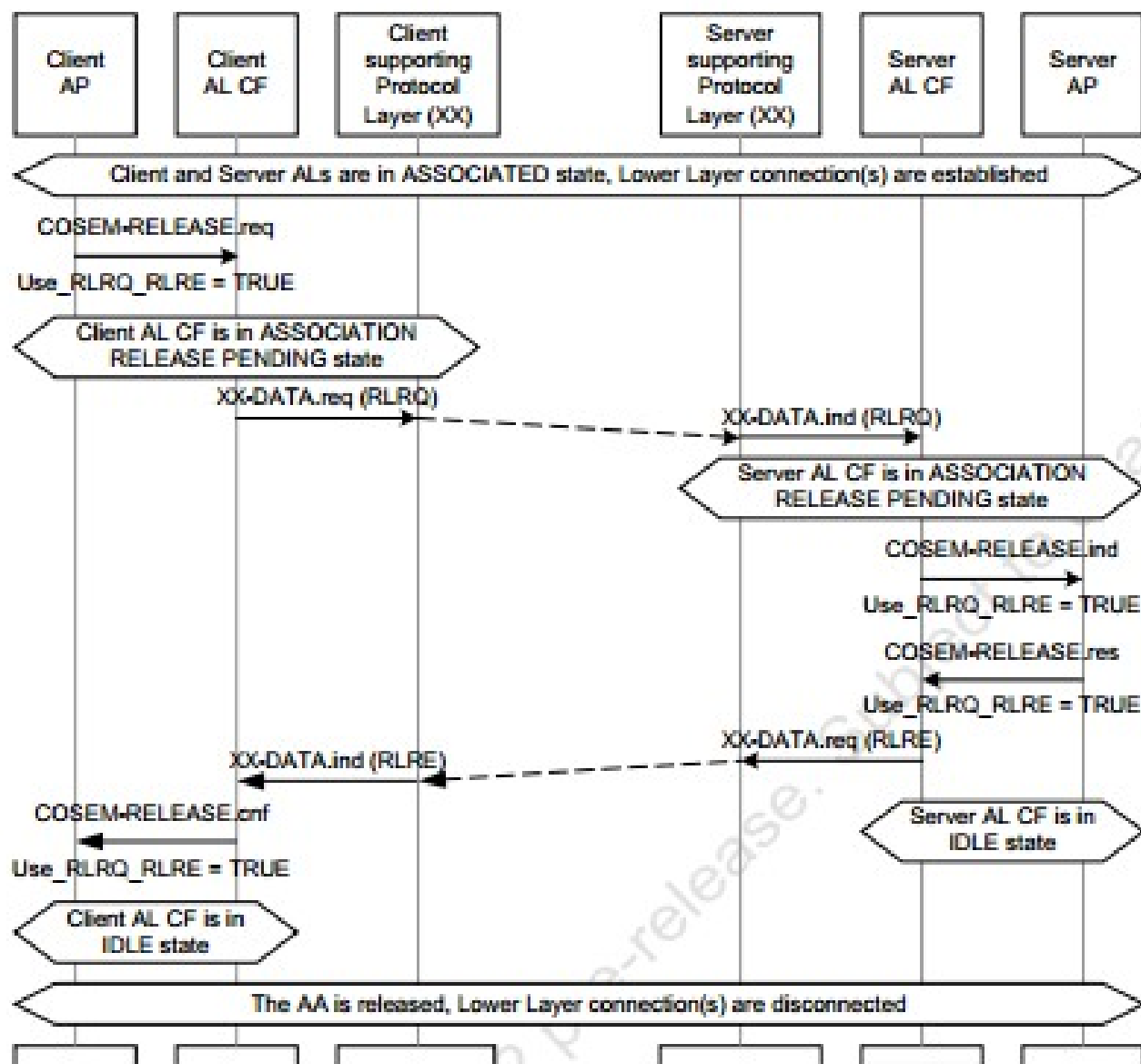
- Protocol for establishment of confirmed application association.
- Repeated COSEM-OPEN service invocations.
- Establishment of unconfirmed application association.
- Pre-establishment application association



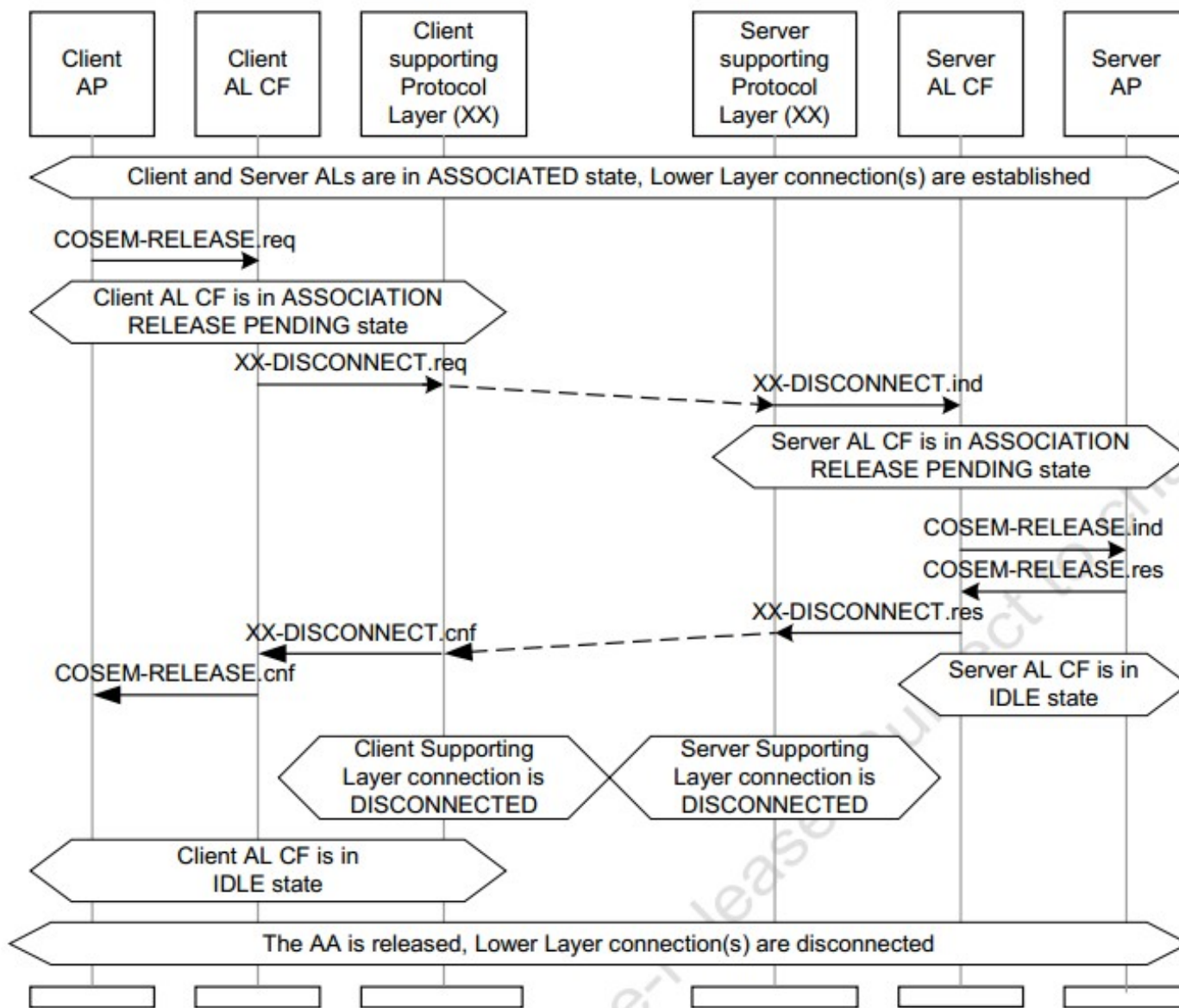
Protocol For Application Associations Release

- Graceful release of an application association.
- Non Graceful release of an application association.

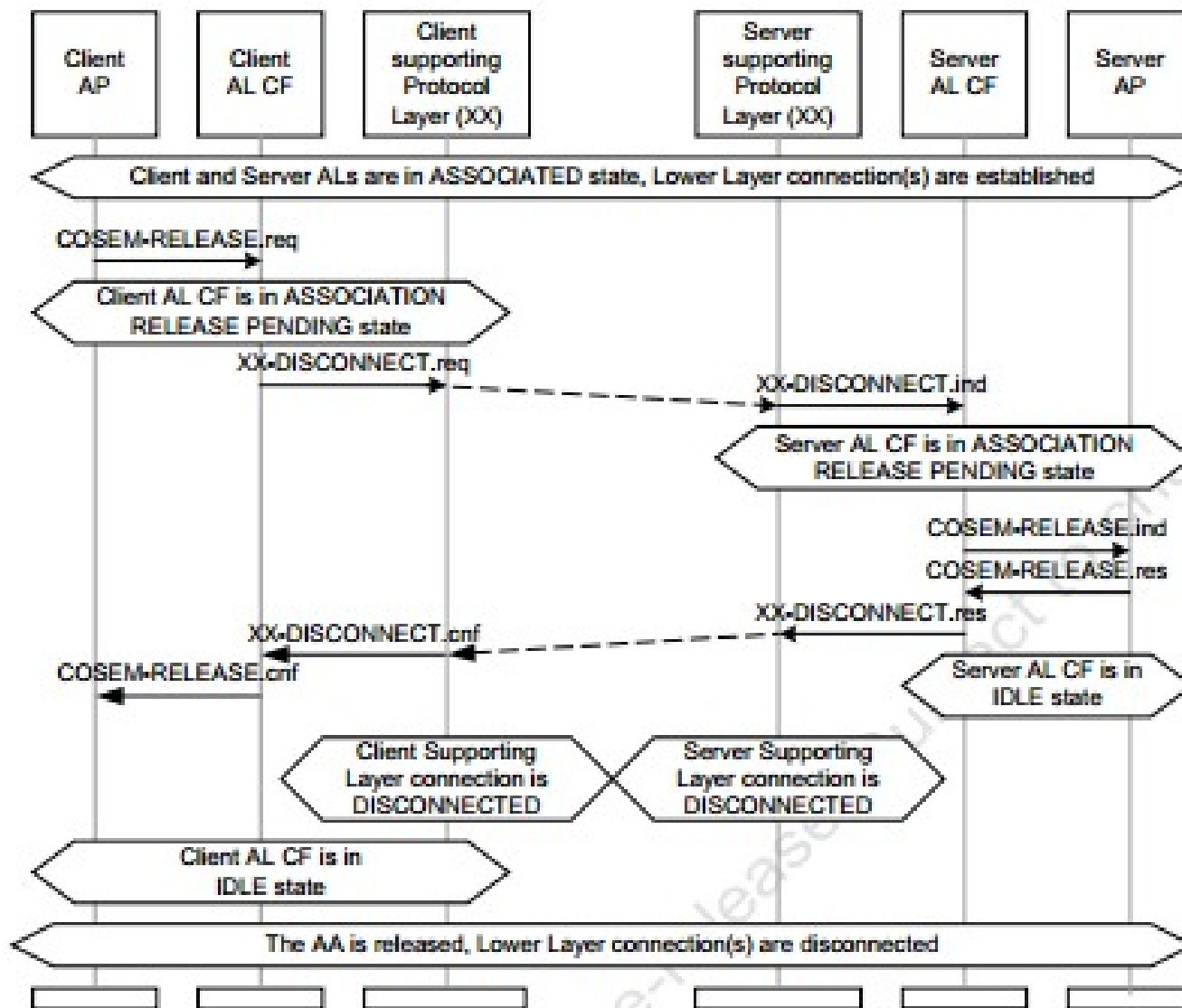
Graceful AA release using the A-RELEASE service



Graceful AA release by disconnecting the supporting layer



Non-graceful release of an application association



THANK YOU