7.3.6.3 Non-graceful release of an application association

Various events may result in non-graceful release of an AA: detection of the disconnection of any lower layer connection (including the physical connection), detecting a local error, etc.

Non-graceful release – abort – of an AA is indicated to the COSEM AP with the help of the COSEM-ABORT.indication service. The Diagnostics parameter of this service indicates the reason for the non-graceful AA release.

Figure 21 shows the message sequence chart for aborting the AA, due to the detection of a physical disconnection.

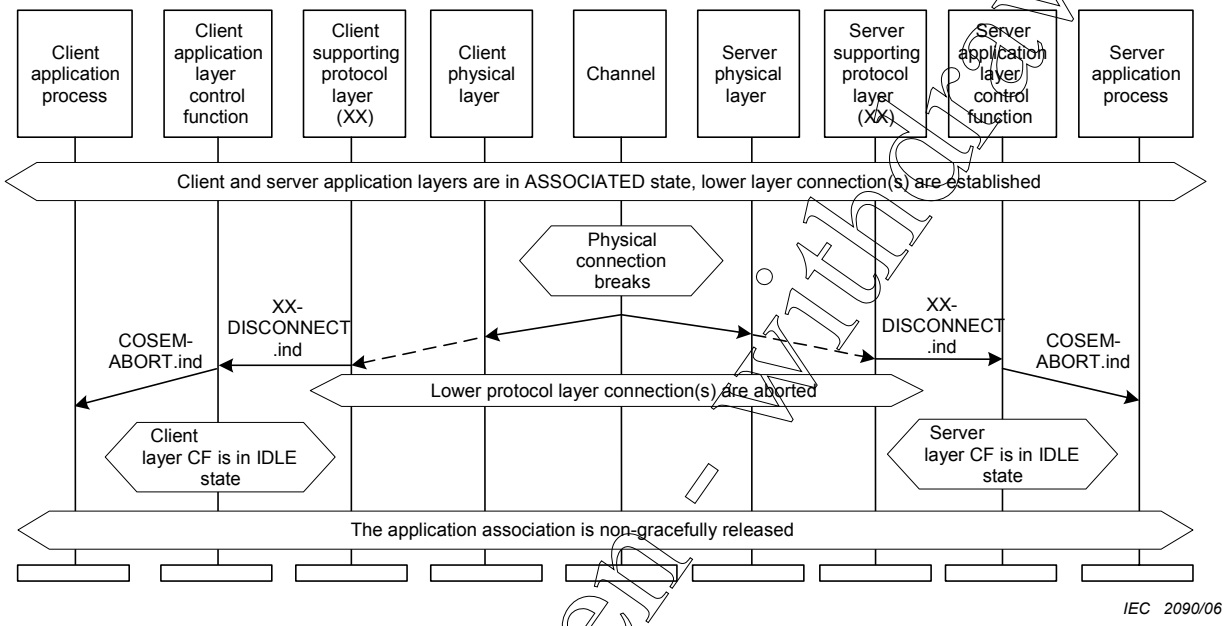

Figure 21 – Aborting an application association following a PH-ABORT.indication

The non-graceful release of AA is not selective: if it happens, all the existing association(s) shall be aborted.

7.3.7 Registered COSEM names

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 the COSEM environment, these objects are assigned by the DLMS User Association, and are specified in this standard.

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

- COSEM application context names (for LN and SN references, without or with cyphering);
- COSEM authentication mechanism names.

7.3.7.1 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[16]. 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)}

where the value of the context_id parameter selects a pre-existing application context.

There are four application context names specified:

COSEM_Application_Context_Name-Logical_Name_Referencing_no_ciphering::=
{joint-iso-ccitt(2) country(16) country-name(756) identified-organisation(5) DLMS-UA(8) application-context(1) context_id(1)}
COSEM_Application_Context_Name-Short_Name_Referencing_no_ciphering::=
{joint-iso-ccitt(2) country(16) country-name(756) identified-organisation(5) DLMS-UA(8) application-context(1) context_id(2)}
COSEM_Application_Context_Name-Logical_Name_with_ciphering::=
{joint-iso-ccitt(2) country(16) country-name(756) identified-organisation(5) DLMS-UA(8) application-context(1) context_id(3)}
COSEM_Application_Context_Name-Short_Name_Referencing_with_ciphering::=
{joint-iso-ccitt(2) country(16) country-name(756) identified-organisation(5) DLMS-UA(8) application-context(1) context_id(4)}

The meaning of these COSEM application contexts is:

- there are two ASEs present within the application-entity invocation, the ACSE and the xDLMS-ASE;
- the xDLSM-ASE is as it is specified in 61134-4-41[17];
- the transfer syntax is A-XDR;
- context_id(1): logical name referencing, no ciphering used;
- context_id(2): short name referencing, no ciphering used;
- context_id(3): logical name referencing, ciphering used;
- context_id(4): short name referencing, ciphering used.

NOTE Ciphering algorithms are not defined in this standard.

---

16 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.

17 With the COSEM extensions to DLMS, see Annex A.

In order to successfully establish an AA, the AARQ and AARE APDUs should carry one of the “valid” values in their application-context-name fields. However, when the server does not accept the proposed application context, the application-context-name field in the AARE response may optionally be different from the application-context-name received in the AARQ. The server may indicate in this way to the client one of the application-context-names supported by the server.

7.3.7.2 COSEM authentication mechanism names

Authentication is one of the security aspects addressed by the COSEM specification. In order to provide different levels of security for authentication support, COSEM specifies three levels of authentication securities:

- 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 uses the authentication feature of the connection-oriented ACSE and for high level authentication, also the methods of the Association LN/SN objects. The process of LLS and HLS authentication is described in IEC 62056-62. To identify the authentication mechanism used, the following object identifiers for authentication mechanism names are specified:

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 parameter selects one of the specified security mechanisms. There are five authentication mechanism names specified:

default-COSEM-lowest-level-security-mechanism-name[18]::= {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) authentication_mechanism_name(2) mechanism_id(0)}
default-COSEM-low-level-security-mechanism-name::= {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) authentication_mechanism_name(2) mechanism_id(1)}
default-COSEM-high-level-security-mechanism-name::= {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) authentication_mechanism_name(2) mechanism_id(2)} NOTE The mechanism name for high-level security starts from 2 and it is registered by the DLMS UA. Mechanism_id(2) is manufacturer specific.
default-COSEM-high-level-security-mechanism-name_using_MD5::= {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) authentication_mechanism_name(2) mechanism_id(3)}
default-COSEM-high-level-security-mechanism-name_using_SHA-1::= {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS-UA(8) authentication_mechanism_name(2) mechanism_id(4)}

The mechanism name element of the AARQ/AARE APDU is present only, when authentication is used. See 7.3.3.

---

18 This mechanism is used for the mandatory AA between a public client and the management logical device in a physical metering device.

7.4 Protocol for data communications

All data communication services are client/server services, except the EventNotification services. Data communication is always initiated by the client by invocation of GET/SET/ACTION.request services. Upon invocation of any of these services, the client application layer protocol machine builds the corresponding APDU and sends it to the peer server application layer.

Data communication service requests can be invoked in a confirmed or an unconfirmed manner. When a service is invoked in a confirmed manner, the server shall respond to the request, otherwise no application level confirmation is expected. See 7.4.1.1.

Unconfirmed services might be invoked in two different ways: individually addressed or broadcast (multicast). See 7.4.1.2.

There is a fourth, non-client/server data communications service supported, the Event Notification service. By requesting this service, the server AP is able to send an unsolicited notification of the occurrence of an event to the remote client. See 7.4.1.3.

7.4.1 Protocol for the xDLMS services using LN referencing

7.4.1.1 Protocol for confirmed services

For confirmed data communication services, the following service primitives are available:

- GET (.request/.indication/.response/.confirm);
- SET (.request/.indication/.response/.confirm);
- ACTION (.request/.indication/.response/.confirm).

GET and SET services are referencing attribute(s) of COSEM interface object instances. The ACTION service is referencing a method of a COSEM interface object instance (e.g. capture a pre-defined set of data). For definition of attributes and methods of COSEM interface classes, see IEC 62056-62.

The COSEM client may invoke the.request primitive of these services in a confirmed manner within a confirmed AA only.

The COSEM server AP, upon the receipt of a data communication service indication, shall check whether the service can be provided or not (validity, client access rights, availability, etc.). If everything is OK, it locally applies the required service on the corresponding ‘real’ object. If a response is required, the COSEM server AP shall generate the appropriate.response message.

Figure 22 shows a complete message sequence chart for a confirmed GET.request service invocation in case of success.

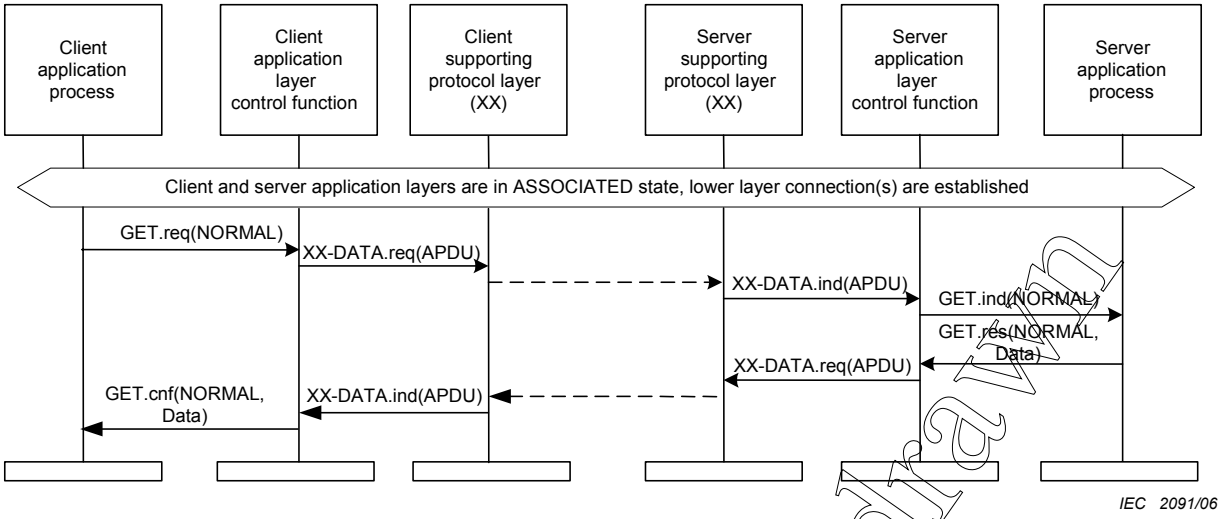

Figure 22 – MSC for a confirmed GET service in case of success

NOTE The message sequence on the figure above applies only if the transferred data does not exceed the supported maximum size of the APDU. In order to be able to transfer longer data with the GET service, COSEM provides an application layer level protocol. In addition, a data link layer level protocol is also available, which is transparent for the application layer. See 7.4.1.8.1.

Figure 23 shows the complete message sequence chart for a confirmed SET service, in case of success.

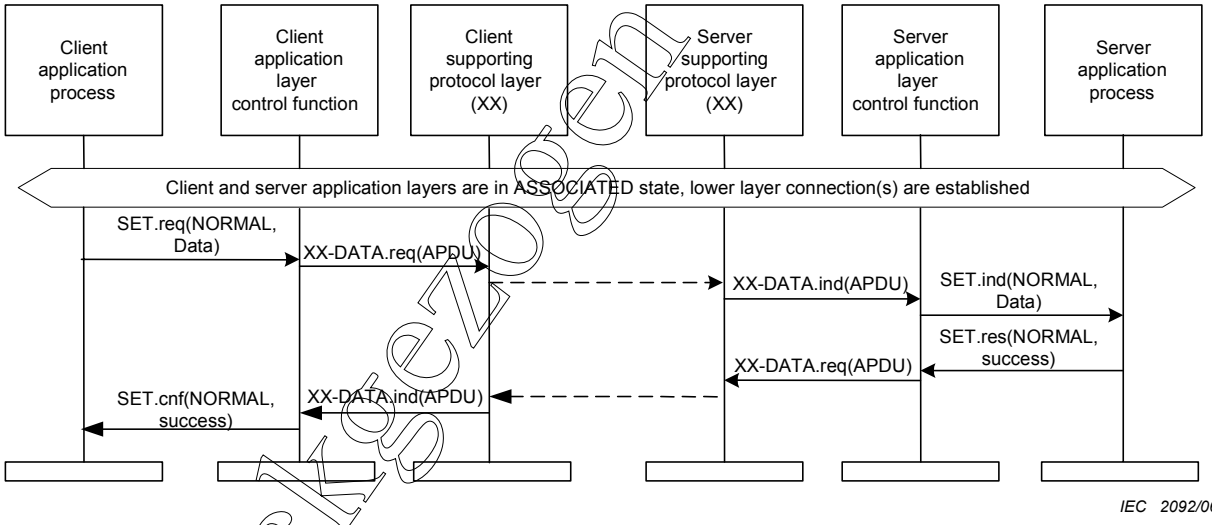

Figure 23 – MSC for a confirmed SET service in case of success

In case of failure, the server – instead of a positive acknowledgement, shown on the above figure – shall send a negative acknowledgement, indicating the reason for the failure, as it is shown in Figure 24.

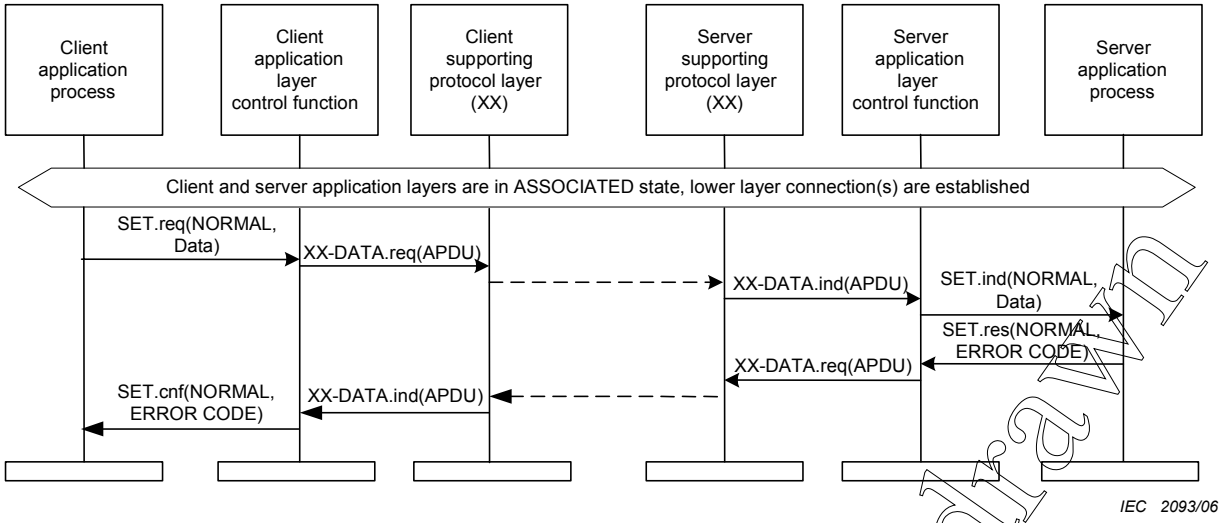

Figure 24 – MSC for the SET service in case of failure

NOTE The message sequence in the above figures applies only if the transferred data does not exceed the supported maximum size of the APDU. In order to be able to transfer longer data with the SET service, COSEM provides an application layer level protocol. This is described in 7.4.1.8.3.

The most complex behaviour is associated with the ACTION service, used for remote invocation of a method of a COSEM interface object in the server. The reason for this complexity is that the invocation of this method may imply data exchange in both client to server and server to client directions, and these data may be too long to fit into one APDU.

Figure 25 illustrates the message sequence chart in the case, when the required service can be granted by the server and the method invocation does not return data.

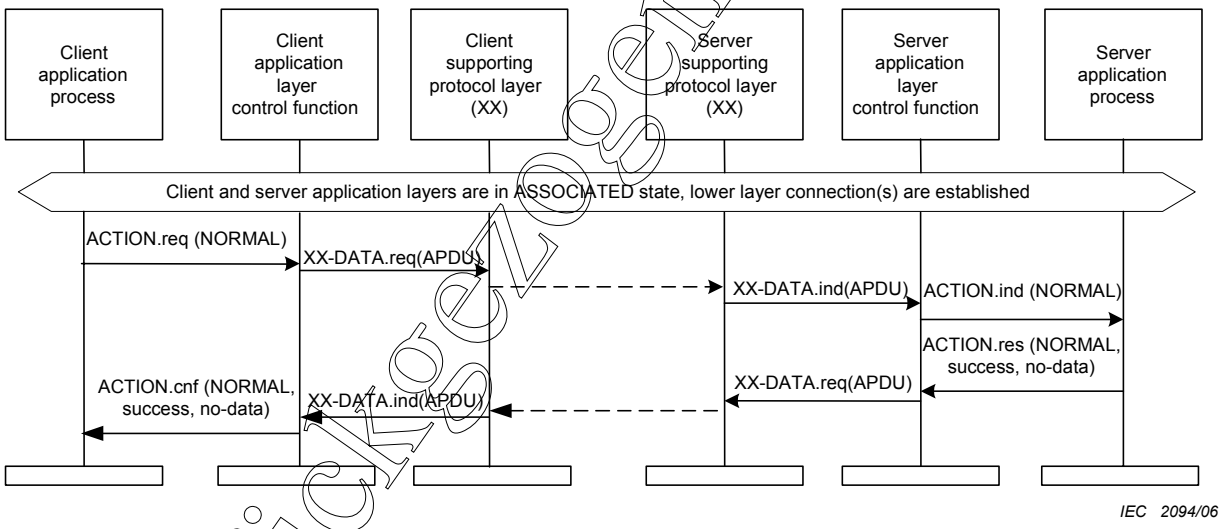

Figure 25 – MSC for the ACTION service (simplest case)

NOTE When either the parameters of the ACTION.request or the ACTION.response service do not fit in one APDU, the protocol defined in 7.4.1.8.4 for transferring long service parameters can be used.

7.4.1.2 Protocol for unconfirmed services

All client/server services may also be invoked in an unconfirmed manner within an established confirmed or unconfirmed AA. The following service primitives are supported:

- GET (.request/.indication);
- SET (.request/.indication);

- ACTION (.request/.indication).

The COSEM client may only invoke these.request primitives when an AA has already been established.

Three different kinds of destination addresses are possible: individual, group or broadcast. Depending on the destination address type, the receiving station shall handle incoming messages differently, as follows:

- XX-PDUs with an individual address of a COSEM logical device. If they are received within an established AA they shall be sent to the addressed COSEM logical device, otherwise shall be discarded;
- XX-PDUs with a group address of a group of COSEM logical devices. These shall be sent to the addressed group of COSEM logical devices. However, the received message shall be discarded if there is no association established between a client and the addressed group of COSEM logical devices;
- XX-PDUs with the broadcast address shall be sent to all addressed COSEM logical devices. However, the received message shall be discarded if there is no association established between a client and the All-station address.

NOTE Unconfirmed AA-s between a client and a group of logical devices are established with a COSEM-OPEN.service with service_class = unconfirmed and a group of logical device addresses (e.g. broadcast address).

7.4.1.3 Protocol for the EventNotification service

This subclause specifies the protocol for the EventNotification.request service of the server application layer, specified in 6.6.3.2.7.

Events in metering equipment, like passing thresholds, fraud detection, or simply a counter overflow generally occur asynchronously to any operation. Depending on the implemented behaviour, the server may want to notify the client immediately.

As in the client/server environment the server is allowed to send information only upon a request from the client, the client may or may not gain knowledge about these events using COSEM client/server type services.

In order to ensure that the server can inform the client about events, a special, non-client/server type service, the EventNotification[19] service is available. As the EventNotification service is not a client-server type service, it may be sent out even when an AA is not established.

Upon invocation of the EventNotification.request service, the COSEM server application layer shall build an EventNotification.request APDU.

The possibilities to send out this APDU depend on the communication profile and the connection status of the lower layers. Therefore, the protocol of the EventNotification service is further discussed in Clause B.2 and Clause B.3.

In any case, in order to notify the client about the detection of an event by the server:

- the server shall use the EventNotification.request service invocation;
- this service invocation shall make the server application layer to build an Event Notification.request APDU;

---

19 When short name referencing is used, the service is called InformationReport at the server side.

- this APDU shall be carried by the supporting layer service at the first opportunity to the client. The service type and the availability of this first opportunity depends on the communication profile used;.
- upon the reception of the EventNotification APDU, the client application layer shall generate an EventNotification.indication[20] service to the COSEM client application;
- by default, event notifications are sent from the management logical device (server) to the management AP (client).

7.4.1.4 Identifying a service invocation: using *Invoke_Id*

A complete confirmed data communication service sequence consists of the exchange of a.request and a.response type message (indicated to the peer protocol layer via the.indication and.confirmation service primitives). In the client/server model, requests are sent by the client and responses are sent by the server. As the client is allowed to send several.requests before receiving the.response for the previous one(s), it is necessary to make a reference in the response message to the corresponding.request message. Otherwise, it is not possible to identify, which.request corresponds to a.response.

The Invoke_Id parameter identifies a.request and the corresponding.response. The value of this parameter is assigned by the client so that each.request primitive issued carries a different Invoke_Id. The server shall copy the Invoke_Id of the received.request message into the corresponding.response message.

The Invoke_Id is not present in the COSEM-OPEN services; these services are identified by their address parameters.

The EventNotification service – as it is not a client/server type service – does not contain Invoke_Id parameter either. There is no corresponding.response service, thus there is no need to use Invoke_Id. See also 8.3.

7.4.1.5 Using priority

COSEM defines two priority levels, 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 shall serve incoming service.requests in the order of their reception (FIFS, First In, First Served[21]). However, it is possible to request to be served first by setting the priority parameter of a.request to HIGH: a.request with priority HIGH shall be served before the previous requests with priority NORMAL. The response primitive shall carry the same priority flag as that of the corresponding.request. Managing priority is a negotiable feature: its support is indicated by BIT 9 of the xDLMS conformance block.

NOTE If the feature is not supported, requests with HIGH priority shall be served with NORMAL priority.

7.4.1.6 Selective access

GET/SET services typically reference the entire attribute of a COSEM interface object. However for certain attributes, selective access to just a part of the attribute may be provided. The part of the attribute is identified by specific selective access parameters.[22] These selective access parameters are defined as part of the attribute specification of the given COSEM interface class specification, see IEC 62056-62.

---

20 At the client side, it is always EventNotification.indication, independently of the referencing scheme (logical name or short name) used at the server side.

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

22 Although the specification of these selection parameters is independent of the referencing method used (LN or SN), the use of these parameters is different for services using logical name (LN) referencing (GET/SET), and services using short name (SN) referencing (read/write). In this subclause selective access for the case of LN referencing is discussed. Selective access with SN referencing is called 'parameterized access', and is discussed in 7.4.2.7.

The selective access specification always starts with an access selector, followed by an access-specific access parameter list. In order to encode the selective access parameters, a 'selective-access-descriptor' type has been specified:

```
Selective-Access-Descriptor::= SEQUENCE
{
access-selector          Unsigned8,
access-parameters        Data
}
```

Using this type, the required parameters for selective access are included in the corresponding LN APDUs as an OPTIONAL field:

```
access-selection          Selective-Access-Descriptor OPTIONAL
```

Selective access is a negotiable feature: its support is indicated by BIT 21 of the xDLMS conformance block.

7.4.1.7 Multiple references in the same service request

7.4.1.7.1 The Attribute_0 reference

GET/SET services typically reference one attribute of a COSEM interface object. The attribute referenced is identified by the value of the COSEM_Object_Attribute_Id parameter.

By convention, attributes are numbered from 1 to n, where Attribute_1 is the logical name of the COSEM interface object. Manufacturers may add proprietary methods and/or attributes to any object, using negative numbers. See also 4.1. of IEC 62056-62.

The value of 0 (zero) for the COSEM_Object_Attribute_Id (Attribute_0)[23] has a special meaning: it references all attributes with positive index (public attributes).

A GET.request service with COSEM_Object_Attribute_Id = 0 requests the value of all public attributes of the referenced object. The response to this request shall be a structure containing the value for all public attributes (data) in the order of their appearance in the given object specification. For attributes to which no access right is granted within the given association, or which cannot be accessed for any other reason, a null_data type NULL value shall be returned.

A SET.request service with COSEM_Object_Attribute_Id = 0 requests to set the value of all public attributes of the referenced object. The data sent with this request shall be a structure, containing for each public attribute, in the order of their appearance in the given object specification, either a value or a null_data type NULL value. The meaning of this NULL value is that the given attribute need not be set.

The response to this request shall be a structure containing the result for each public attribute (data-access-result), in the order of their appearance in the given object specification, indicating the success or failure of the requested SET operation. The response shall be carried by a SET-Response-With-List – type APDU.

Attribute_0 referencing is a negotiable feature: its support for the GET service is indicated by BIT 10, and for the SET service by BIT 8 of the xDLMS conformance block.

[23] The Attribute_0 feature cannot be applied when short name referencing is used.

7.4.1.7.2 Attribute reference list

A complete (LN) reference for an attribute includes the following parameters:

class-id	Cosem-Class-Id,
instance-id	Cosem-Object-Instance-Id,
attribute-id	Cosem-Object-Attribute-Id,
access-selection	Selective-Access-Descriptor **OPTIONAL**

A.request service may contain one such reference or a list of such references. Specification of the APDUs for the different types of.requests is given in 8.6.

7.4.1.8 Transferring long service parameters

7.4.1.8.1 Non-transparent and transparent transfer mechanisms

The service parameters of data communication services are transported by the APDUs, exchanged between the peer layers, in an encoded form. In some cases, the APDU can be longer than what the protocol is able to transmit in one piece. In order to be able to exchange such 'long' data, two transporting mechanisms are available:

a) long data transfer using an application level protocol. This mechanism can be used with any of the specified services (GET, SET and ACTION) and with any protocol profile and is specified in the following subclauses;

b) long data transfer in a transparent manner to the client application. This feature can be used only with lower layer protocols providing segmentation. 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 application layer. As these services are specific to the supporting protocol layer, handling these services is not within the scope of this standard – in other words, no specific application layer services and protocol are specified for this purpose. When the supporting protocol layer supports transparent long data transfer, the server side application layer implementation may be able to manage these services.

7.4.1.8.2 Application protocol for long data transfer with the GET service

Long data transfer with the GET service is specified only for the data in the GET.response service primitive.

The length of the encoded form of service parameters for selective access and/or multiple attribute references in the GET.request service shall not exceed the maximum allowed size of APDUs.

GET.request services shall be of type NORMAL or WITH-LIST. Upon reception of a GET.request, the server AP shall assemble the requested data. If the data fit into one APDU, the server AP shall invoke the GET.response service with NORMAL or WITH-LIST type, with the value(s) of the required attribute(s) as the result parameter.

If the data do not fit into one APDU and block transfer is supported (bit11 of the xDLMS conformance block), the server AP shall send the data in blocks.

First, the data shall be encoded, as if they would fit into one APDU. The result is a series of bytes, $D_1,D_2,D_3,....D_N$. The server shall then assemble a DataBlock-G data structure (see 8.3) with the following contents:

last-block (BOOLEAN)	= FALSE
block-number (Unsigned32)	= 0001
result (IMPLICIT OCTETSTRING)	= the first K bytes of the encoded data ($D_1,D_2,D_3,....D_K$)

This DataBlock-G shall be the first part of the response. The server AP shall invoke the GET.response service with Response_type = ONE-BLOCK, with the Invoke_Id and priority parameters copied from the GET.request invocation received and with the DataBlock-G as result parameter.

Upon reception of this GET.response (signalled as.confirm), the client AP is informed that the response for its request does not fit into one APDU and shall proceed for the long data transfer. It shall store the data contents of the received APDU – (D_1,D_2,D_3,....D_K) – and shall acknowledge the received block by invoking the GET.request service with Request_type = NEXT and with the following parameters:

invoke-id-and-priority = the same as that for the first GET.request;

block-number = the same as the Block-number of the received data block.

When the server receives the acknowledgement, it shall prepare and send the next data block, including D_{K+1},D_{K+2},D_{K+3},....D_L, with block-number = 0002. This exchange of data blocks and acknowledgements shall normally continue until the last Data Block, including D_M,D_{M+1},D_{M+2},....D_N is sent. The last-block (BOOLEAN) parameter of this DataBlock-G sequence shall be set to TRUE and this data block shall not be acknowledged by the client. After the reception of the last data block, the long data transfer with the GET service is completed.

Figure 26 shows an example for the case, when the requested data can be sent in three parts, and the transfer is not aborted.

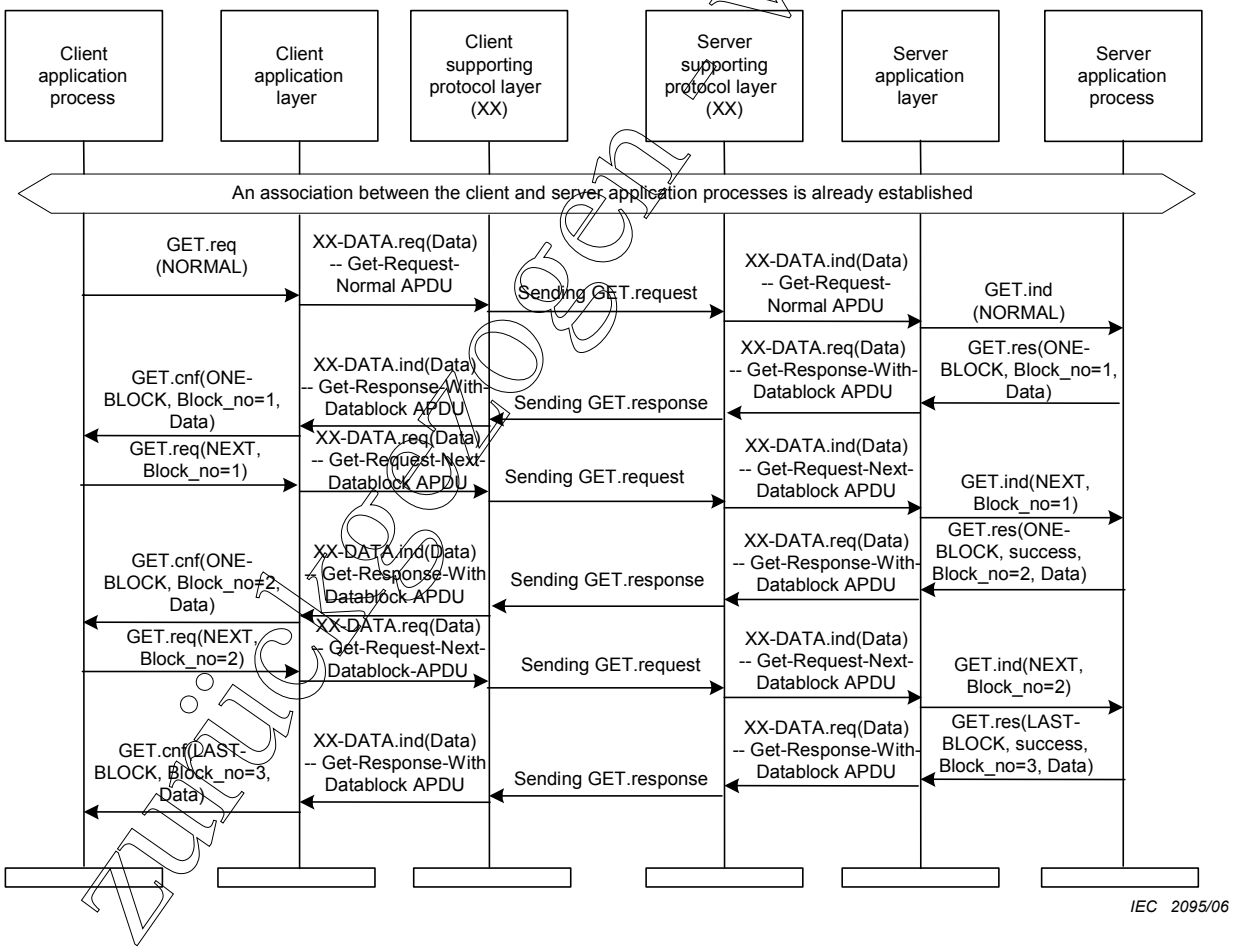

Figure 26 – Long data with the GET service in three data blocks