STS Association

STS 101-2

Edition 1.6

Aug 2021

Standard Transfer Specification – Interface specification – Physical layer protocol for a two-way virtual token carrier for remote connection over DLMS/COSEM

CONTENTS

Revision History

Edition	Clause	Date	Change details
1.1	general	March 2015	Various editorial changes made
1.2	Table 1	April 2015	Changed STS_COS_TC reference in Table 1
1.2	Normative references	April 2015	Added STS 202-3 reference
1.3	Table 11	May 2016	Changed token enumerated value 3 “o "Token Accep”ed". Changed to new Logo.
1.4	General technical changes	Nov 2016	Added note to 7.4.2 to indicate format failure conditions. Table –1 - changed bullet 4 and 8 descriptions. 7.2–1 - Added definition for tokendatatype = 0. Updated document as per comments received during CDV stage.
1.5	technical	January 2021	Added note to Table 4 Added bit-string values in Tables 6-9 Corrected example in A.3
			Added Note2 to 7.2.7, and Note1 to 7.2
	technical		Added text 7.4.2 to indicate that processing is to stop after detection of the first token failure.
	technical		Changed all occurrences of ‘DLMS Blue Book’ to ‘IEC62056-6-2’.
	3.4		Added clause on naming conventions
	4.2		Added Figure 3
	4.3		Added cluse on Token behaviour
	6.1, 6.2		Added clauses for token description request, renumbered subsequent clauses
	6.1.1		Added sub-clause explaining the processing sequence
	6.2.1		Added clause on Use of TokenDescription Attribute
	general	March 2021	Editorial polishing to STS template styles and alignment of element names with IEC 62055-41 and IEC 62056-6-2
1.5		May 2021	Published
1.6	Table10, and 7.4.1 Table 23, 24	Aug 2021	Removed references to TokenCarrierType and IDRecordExpired in the IDRecord Table 23 and 24 corrected.

STANDARD TRANSFER SPECIFICATION ASSOCIATION

_____

STANDARD TRANSFER SPECIFICATION –

Interface specification – Physical layer protocol for a two-way virtual token carrier for remote connection over DLMS/COSEM

FOREWORD

1) The Standard Transfer Specification Association (STSA) is a worldwide organization for standardization comprising all members of STSA. The object of STSA is to develop, maintain and promote international use of the Standard Transfer Specification (STS). To this end and in addition to other activities, STSA publishes Standards, Technical Specifications, Technical Reports, Codes of Practice and Guides (hereafter referred to as "STSA Publication(s)"). Their preparation is entrusted to technical working groups; any STSA member interested in the subject dealt with may participate in this preparatory work. STSA collaborates closely with the International Electrotechnical Commission (IEC) in accordance with conditions determined by agreement between the two organizations. As such STSA performs the role of Registration Authority of IEC 62055-41, IEC 62055-51 and IEC 62055-52 on behalf of IEC.
2) The formal decisions or agreements of STSA on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each working group has representation from all interested STSA members.
3) STSA Publications have the form of recommendations for international use and are accepted by STSA Board of Directors in that sense. While all reasonable efforts are made to ensure that the technical content of STSA Publications is accurate, STSA cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
5) STSA provides attestation of conformity. Independent testing bodies provide conformity assessment services and recommendations to STSA Board of Directors who provides conformance certificates and access to STSA marks of conformity.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to STSA or its directors, employees, servants or agents including individual experts and members of its technical working groups for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this STSA Publication or any other STSA Publications.
8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this STSA Publication may be the subject of patent rights. STSA shall not be held responsible for identifying any or all such patent rights.

Standard Transfer Specification STS 101-2 has been prepared by working group 8.

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

CDV	Report on voting
STS 101-2 /CDV	21/06/2021

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with STSA Directive STS 2100-1.

INTRODUCTION

The Standard Transfer Specification (STS) is a secure message protocol that allows information to be carried between point of sale (POS) equipment and payment meters and it caters for several message types such as credit, configuration control, display and test instructions.

There is an emerging need in the global marketplace to harmonize existing standards in general and STS in particular in order to meet the developing Smart Meter and Smart Grid requirements.

IEC 62055-51 presently covers token carriers that require manual entry at the end device such as a prepayment meter.

DLMS/COSEM is a widely accepted protocol standard being used by smart metering devices, enabling bi-directional communication between a head end system client and an end device server. Included in the COSEM suite are classes such as Account, Credit, Charge, TokenGateway, Tariff, Calendar and Switch, which are especially suited for payment metering functions.

The TokenCarrier is the physical medium used to transport information from a POS or the management system to the payment meter, or from the payment meter to the POS or management system. This document specifies a virtual token carrier as embodied in a remote connection between a management device client and a payment meter server.

This document specifies a new TokenCarrier that allows for the transportation of STS tokens over a DLMS/COSEM channel, while simultaneously supporting the existing token carriers such as defined in IEC 62055-51 and IEC 62055-52.

The STS message is an ASN.1 message encoded using the A-XDR encoding rules, and is contained within the DLMS/COSEM message block as a variable length Octet string.

STANDARD TRANSFER SPECIFICATION –

Interface specification – Physical layer protocol for a two-way virtual token carrier for remote connection over DLMS/COSEM

1 Scope

This document specifies the requirements for a new TokenCarrierType (STS_COSEM_TC) for the transfer of tokens using standard DLMS/COSEM client/server protocols and a selection of COSEM interface classes.

The normative part of this specification is limited to the translation from the STS application layer protocol into the DLMS/COSEM client application layer protocol and does not concern itself with details of lower layers of the communication protocol stack, which is the subject of other standards.

Use cases and domain objects are presented as informative only to illustrate how the STS_COSEM_TC can be used in a practical implementation. It also serves as a reference model to test behaviour of the TokenCarrier.

This document specifies the format of the TokenCarrier to be produced by a POS (client) or consumed by an STS-compliant payment meter (Server) in an integrated STS-COSEM system, typically over a remote connection. It is complementary to the application layer protocol specified in IEC 62055-41 and should be used in conjunction with that standard.

It is intended for use across a range of payment meters developed by different manufacturers and to ensure compatibility between these products and other client devices.

A utility should be able to request STS-COSEM integration by including this document by reference in their Companion Specification, and specifying the concrete TokenGateway InstanceID required on their meters.

Unlike other STS token carriers this document specifies a two-way carrier:

- There is a path from the POS to the Meter
- There is a return path from the Meter to the POS, to convey the result of processing.

The main design objective in establishing the protocol has been the requirement to integrate the transfer of data to and from an STS compliant payment meter using the existing DLMS/COSEM protocol, which is today widely used in the Smart Meter market. The protocol specified in this document makes use of the DLMS/COSEM protocol, and specifies a data format for inclusion of an STS payload within this DLMS/COSEM protocol.

2 Normative references

IEC 62055-41 Ed3, *ELECTRICITY METERING – PAYMENT SYSTEMS – Part 41: Standard transfer specification (STS) – Application layer protocol for one-way token carrier systems*

IEC 62056-5-3 Ed 3.0, *ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 5-3: DLMS/COSEM Application layer*

IEC 62056-6-2 Ed 3.0, *Electricity Metering Data Exchange – The DLMS/COSEM Suite – Part 6-2: COSEM Interface classes*

IEC 62056-6-1Ed 3, *Electricity Metering Data Exchange – The DLMS/COSEM suite – Part 6 - 1: COSEM Object Identification System (OBIS)*

IEC 61334-6 Ed 1:2000, *Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule*

ITUT –TREC X690 2008-11*, Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

3 Terms, definitions and abbreviations

3.1 Definitions

For the purposes of this standard, the definitions and terms given in IEC 62055-41 shall apply.

3.2 Abbreviations

ASN1	Abstract Syntax Notation One
A-XDR	Adapted External Data Representation
BER	Basic Encoding Rules of ASN.1
COSEM	Companion Specification for Energy Measurement
DKGA	DecoderKeyGenerationAlgorithm
DLMS	Device Language Message Specification
EA	EncryptionAlgorithm
IEC	International Electrotechnical Commission
KRN	KeyRevisionNumber
SGC	SupplyGroupCode
STS	Standard Transfer Specification
TCT	TokenCarrierType
TDT	TokenDataType
TI	TariffIndex
TID	TokenIdentifier
UC	Use Case
VTC	VirtualTokenCarrier

3.3 Numbering conventions

In this International Standard the representation of numbers in binary strings uses the convention that the least significant bit is to the right, and the most significant bit is to the left.

Numbering of bit positions start with bit position 0, which corresponds to the least significant bit of a binary number.

Numbers are generally in decimal format, unless otherwise indicated. Any digit without an indicator signifies decimal format.

Binary digit values range from 0-1.

Decimal digit values range from 0-9.

Hexadecimal digit values range from 0-9, A-F and are indicated by “hex” or by a preceding “0x”.

3.4 Naming conventions

The words “token”, “tokendata”, and “data” have meanings in STS standards that are distinct from their use in DLMS standards. In this specification these words may refer to an STS token as specified in the STS standard (IEC 62055-41) or a data input to a TokenGateway as specified in IEC 62056-6-2, depending on context:

- From a DLMS TokenGateway perspective a “token” is the “data” input to the “enter” method, which is conveyed as an opaque field within a DLMS/COSEM ASN.1-encoded message;
- From an STS101-2 perspective the “data” input to the “enter” method is an ASN.1-encoded RequestMessage (see 7.3.3);
- A DLMS/COSEM “token” is thus synonymous with an STS101-2 “RequestMessage”, and distinct from an STS 20-digit numeric token carrier or the 66-bit TokenData in the TCDU as defined in IEC 62055-41;
- Similarly the “token_status” structure returned by the enter() method is defined in IEC 62056-6-2. The word “token” in the context of the status_code field must be understood as pertaining to the DLMS/COSEM token. The data_value bit-string is an ASN.1-encoded “DataValue” (see 7.4.1).

4 Requirements

4.1 Domain entities

The reference diagram shown in Figure 1 is a simplified function class diagram extracted from IEC 62056-5-3 and IEC 62056-6-2 (shown in green) in association with relevant STS entities extracted from IEC 62055-41 (shown in yellow).

Information flow is between the Head-endSystemApplicationProcess (HES_AP) and the EndDeviceApplicationProcess (ED_AP).

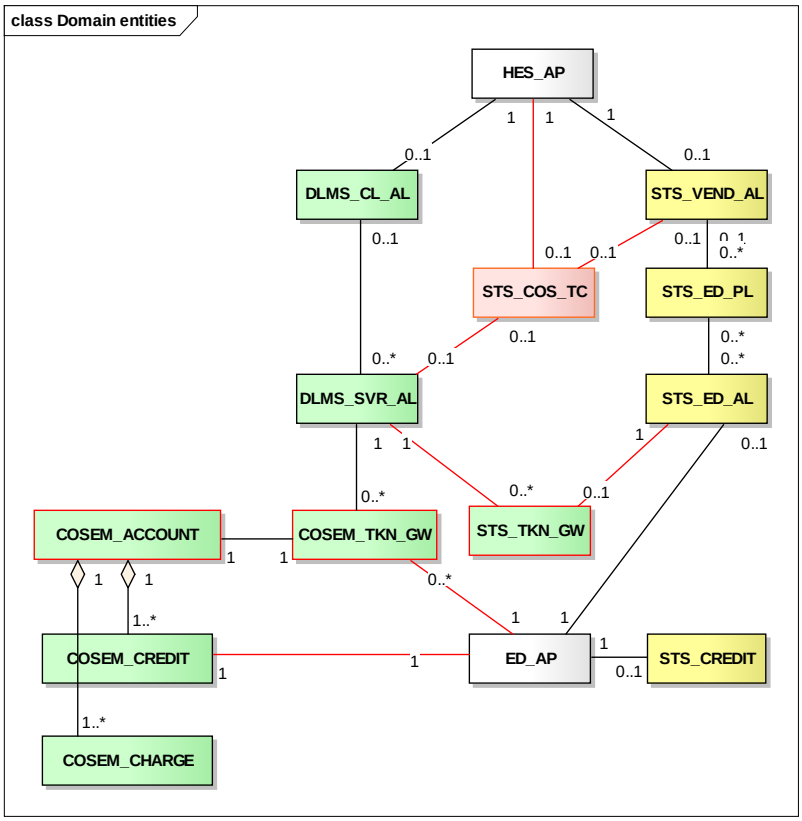

Figure 1 – Domain entities and associations

The specifications for the classes are given in Table 1.

Table 1 – Domain entity specifications

Entity	Context	Specification
COSEM_ACCOUNT	COSEM Account interface class	IEC 62056-6-2
COSEM_CHARGE	COSEM Charge interface class	IEC 62056-6-2
COSEM_CREDIT	COSEM Credit interface class	IEC 62056-6-2
COSEM_TKN_GW	COSEM TokenGateway interface class	IEC 62056-6-2
DLMS_CL_AL	DLMS client application layer	IEC 62056-5-3
DLMS_SVR_AL	DLMS server application layer	IEC 62056-5-3
ED_AP	End device application process	Application-specific
HES_AP	Head-end system application process	Application-specific
STS_COS_TC	STS/DLMS/COSEM TokenCarrier	STS101-2
STS_CREDIT	STS credit register	IEC 62055-41
STS_ED_AL	STS end device application layer	IEC 62055-41
STS_ED_PL	STS end device physical layer	IEC 62055-41
STS_TKN_GW	COSEM STS TokenGateway	IEC 62056-6-2
STS_VEND_AL	STS vending system application layer	IEC 62055-41

4.2 Message interchange

Figure 2 and Figure 3 show reference use case sequence diagrams, from which the STS_COS_TC interface specification is derived.

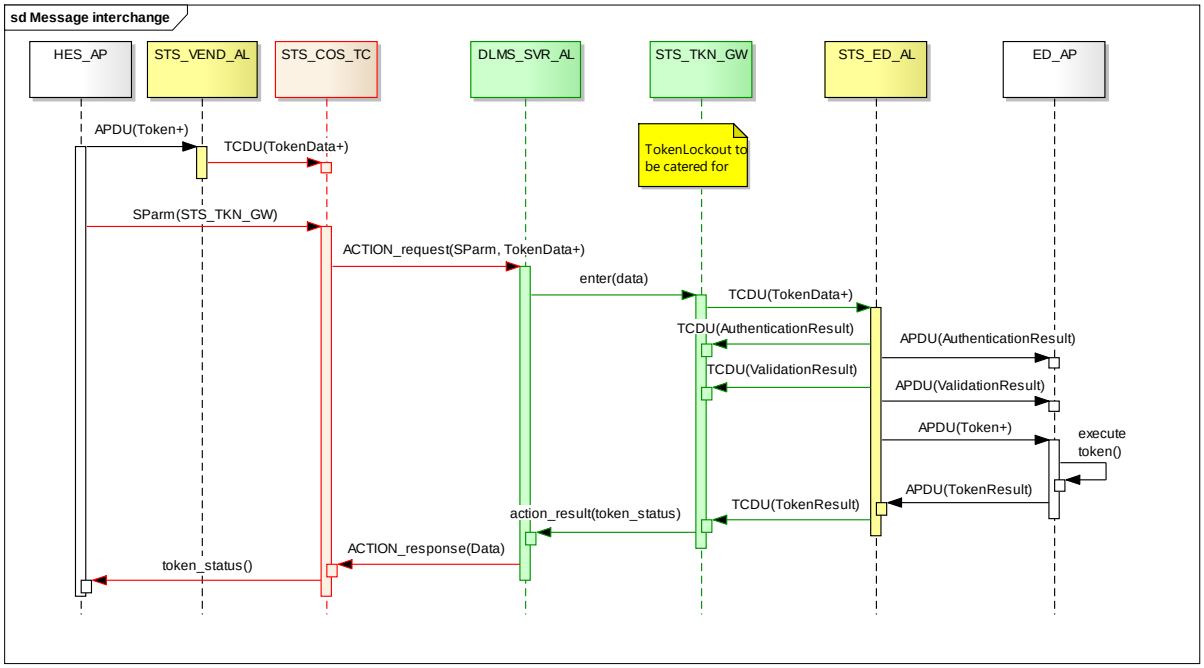

Figure 2 – Message interchange for STS_COS_TC interface – enter(data) method request

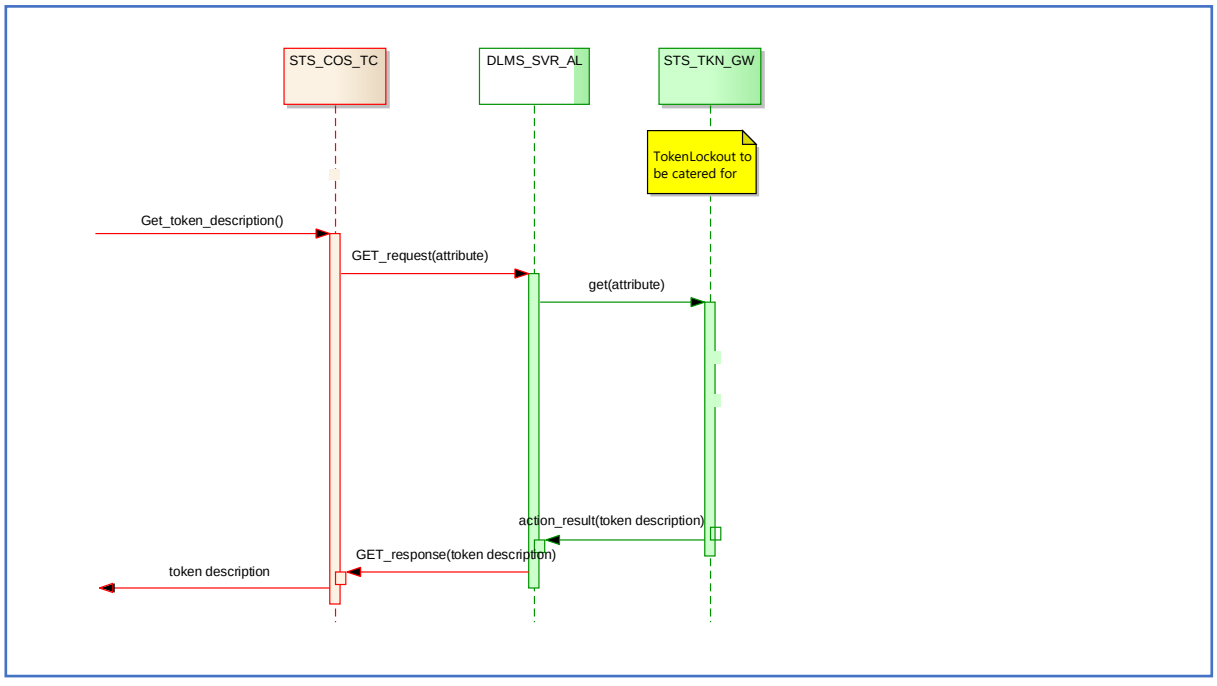

Figure 3 – Message interchange for STS_COS_TC interface – get token description

The message specifications are given in Table 2.

Table 2 – Message interchange specifications

Entity	Context	Specification
APDU(Token+)	ApplicationProtocolDataUnit containing the token plus other data elements	IEC 62055-41 clause 6.1 IEC 62055-41 clause 7.1
TCDU(TokenData)	TokenCarrierDataUnit containing the 66 bit token data	IEC 62055-41 clause 6.4
TCDU(AuthenticationResult)	TokenCarrierDataUnit containing the result from authentication check	IEC 62055-41 clause 7.2
TCDU(ValidationResult)	TokenCarrierDataUnit containing the result from validation check	IEC 62055-41 clause 7.2
TCDU(TokenResult)	TokenCarrierDataUnit containing the result from token execution	IEC 62055-41 clause 7.2
Sparm(STS_TKN_GW)	Service parameters for the DLMS/COSEM TokenGateway interface class	IEC 62056-6-1 IEC 62055-41 clause 6.8
ACTION_request(Spar, TokenData)	DLMS message construct from DLMS/COSEM Client to Server containing Spar plus TokenData	IEC 62056-5-3 clause 6.8
ACTION_response(token_status)	DLMS message construct from DLMS/COSEM Server to Client containing token_status as returned from COSEM TokenGateway interface class	IEC 62056-6-2 IEC 62056-5-3 clause 6.8
enter(data)	Specific method to be used for COSEM TokenGateway to load the TokenData contained in data	IEC 62056-6-2
return(action_result)	COSEM TokenGateway returns the result from the enter method	IEC 62056-6-2
token_status()	Compound data construct as returned from COSEM TokenGateway interface class	IEC 62056-6-2

Note: Server firmware processes start at DLMS_SRV_AL in Figure 2.

4.3 TokenGateway behaviour

STS_TKN_GW is an instance of the DLMS/COSEM ‘Token gateway’ interface class for which behaviour of the enter() method is defined by this standard (STS101-2). Specifically the enter() method:

1) Receives ‘data’ which it parses as an ASN.1-encoded RequestMessage;
2) Extracts STS tokens from the RequestMessage and submits them in order to the STS Application Layer (STS_ED_AL), stopping if a token is rejected;
3) Formats the result from the STS Application Layer into an ASN.1-encoded DataValue;
4) Returns the ‘token_status’ structure which includes the DataValue (as the ‘data_value’ field).

Further detail is given in 6.4

5 POSToTokenCarrierInterface : Physical Layer Protocol

In practice the token is entered into the payment meter by means of, for example, a keypad. A client device could also be a mobile HHU that connects to the payment meter by means of a direct local connection, but it is also possible for the connection to be extended to a remote management system by means of suitable interposing modem devices linked over any appropriate communications medium.

The connection is not within the scope of this specification. However, the format for the data in such extended remote connection is specifically covered in this standard. Additional information on bit-string A-XDR encoding may be found in DLMS UA 1001-2.

6 TokenCarrier

6.1 General

The TokenCarrier is a message in ASN.1 format in compliance with ITUT –TREC X690 2008-11 encoded with A-XDR encoding rules as per IEC61334-6.

The POS shall send the token(s) to the meter by encoding them into the TokenCarrier as described in this standard, then invoking methodId=1 (enter) on the STS TokenGateway.

The TokenCarrier should have forward compatibility (a meter or POS implementation conforming to this standard should be able to at least recognise - and possibly accept - future versions of the TokenCarrier) and backwards compatibility (implementations conforming to a future version of this standard should be able to recognise and accept previous versions of the TokenCarrier).

One approach to achieve such compatibility is to add to each message (request, response) at least one field that is Reserved for Future Use (RFU).

There are several possibilities for the TokenCarrier as outlined in the following clauses.

6.2 GET_request (POS to Meter)

The GET_request to extract the STS token description is according to IEC 62056-6-2 for the COSEM TokenGateway class.

6.3 GET_response (Meter to POS)

The GET_response to extract the STS token description is according to IEC 62056-6-2 for the COSEM TokenGateway class.

The token_description attribute is specified as an array of octet-string type token_description_elements that contain a description of the most recently processed token. In the STS context, the token_description attribute shall comprise of the following token_description_elements:

```
token_description :: = {
        sts_token_class:        octet-string,
        sts_token_subclass:     octet-string,
        sts_token_id:           octet-string,
        sts_token_amount:       octet-string,
}
```

Where:

- sts_token_class and sts_token_subclass represent the STS token classes and sub-classes as given in tables 14 and 15 in IEC 62055-41 and are formatted as the hexadecimal equivalent of ASCII characters as shown in Table 3, Table 4, Table 5 and Table 6.

Table 3 – ASCII/Hex encoding for token classes 0, 1 and 2

Token Class	ASCII representation	sts_token_class representation (Hexadecimal equivalent)
0	CREDIT TRANSFER	435245444954205452414E53464552
1	NON METER SPECIFIC MANAGEMENT	4E4F4E204D4554455220535045434949464943204D414E4147454D454E54
2	METER SPECIFIC MANAGEMENT	4D4554455220535045434949464943204D414E4147454D454E54

Table 4 – ASCII/Hex encoding for token class 0 and sub-classes 0 - 7

Class 0		
Token SubClass	**ASCII representation**	**sts_token_subclass representation (Hexadecimal equivalent)**
0	ELECTRICITY	454C454354524943495459
1	WATER	5741544552
2	GAS	474153
3	TIME	54494D45
4	ELECTRICITY CURRENCY	454C4543545249434954204355525245 4E4359
5	WATER CURRENCY	5741544552204355525245 4E4359
6	GAS CURRENCY	47415320435552524 54E4359
7	TIME CURRENCY	54494D452043555252454E4359

Table 5 – ASCII/Hex encoding for token class 1 and sub-classes 0 - 1

Class 1		
Token SubClass	**ASCII representation**	**sts_token_subclass representation (Hexadecimal equivalent)**
0	INITIATE METER TEST 2DGT	494E495449415445204D4554455220544553542032444754
1	INITIATE METER TEST 4DGT	494E495449415445204D4554455220544553542034444754

Table 6 – ASCII/Hex encoding for token class 2 and sub-classes 0 - 9

Class 2		
Token SubClass	**ASCII representation**	**sts_token_subclass representation (Hexadecimal equivalent)**
0	SET MAXIMUM POWER LIMIT	534554204D4158494D554D20504F574552204C494D4954
1	CLEAR CREDIT	434C45415220435245444954
2	-	
3	SET 1ST DECODER KEY	53455420315354204445434F444552204B4559
4	SET 2ND DECODER KEY	53455420324E44204445434F444552204B4559

5	CLEAR TAMPER CONDITION	434C4541522054414D50455220434F4E444954494F4E
6	SET MAX POWER PHASE POWER UNBALANCED LIMIT	534554204D415820504F5745522050484153452050 4F57455220554E42414C414E434544204C494D4954
7	-	
8	SET 3RD DECODER KEY	53455420335244204445434F444552204B4559
9	SET 4TH DECODER KEY	53455420345448204445434F444552204B4559

- sts_token_id represents the TokenIdentifier as specified in 6.3.5 of IEC 62055-41 and formatted as hexadecimal. Where the TokenIdentifier is not applicable to a particular sts_token_subclass then its value shall be null-data as defined in IEC 62056-6-2.
- sts_token_amount represents the value of the Amount field carried in class 0 tokens as specified in 6.2.2 of IEC 62055-41 after it has been processed and formatted into data type of the current_credit_amount attribute of the Credit interface class specified in IEC 62056-6-2. The processed value is formatted as the hexadecimal equivalent of ASCII characters. Where the Amount field is not applicable to a particular sts_token_subclass then its value shall be null-data as defined in IEC 62056-6-2.

Example 1 – Class 0 tokens

Amount: 10.00 kWh

TID: 120,355

token_description :: = {

435245444954205452414E53464552,

454C454354524943495459,

01D623,

31302E3030

}

Example 2 – Class 1 tokens

Representation of token_description for 2-digit manufacture code test token.

token_description :: = {

4E4F4E204D4554455220535045434946494320 4D414E4147454D454E54,

494E495449415445204D4554455220544553542032444754,

null-data,

null-data

}

Example 3 – Class 2 tokens

Representation of token_description for Set1stSectionDecoderKey token..

token_description :: = {

4D45544552205350454349464943204D414E4147454D454E54,

534554203153542044454 34F444552204B4559,

null-data,

null-data

}

6.4 ACTION_request (POS to Meter)

The request must have the following fields (derived from IEC 62055-41 APDU/TCDU):

- TokenDataType, an integer in the range (0-99);
- IDRecord, an optional, printable string of 35 digits;
- PRNRecord, an optional, printable string of variable length;
- One or more Tokens (Octet String) in binary or Hex.

This standard requires that the enter() method shall have behaviour equivalent to the following:

1) Initialise all fields of DataValue to zero (i.e. no bits set). Whenever this enter() method returns, this DataValue is ASN.1-encoded to create the token_status.data_value field;
2) Parse “data” as an ASN.1 RequestMessage. If the data cannot be decoded or understood successfully then: return with status_code = 4 (Token data type failure) and current data_value;
3) If RequestMessage.idrecord is present then: check the “idrecord” and set DataValue.idRecord (type IDRecordResult). Do not return, even if the IDRecord is not OK;
4) Check the RequestMessage.tokendatatype; if not supported by this STS_TKN_GW implementation, or the corresponding token data is missing or not well-formed (e.g. tokendatatype = 0 but the RequestMessage.tokens sequence is empty) then: set DataValue.token (type TokenResult) bit 10 (invalidTokenDataType), and return with status_code = 7 (Token Execution Result Failure) and current data_value;
5) Iterate over the RequestMessage.tokens sequence in order, doing the following for each token:
    a) Set the bit in DataValue.tkf (type TokenFailure) that corresponds to this token (i.e. set bit ‘token1Failure(0)’ for the first token, or bit ‘token2Failure(1)’ for the second token, etc.);
    b) Pass the token to STS_ED_AL which should (according to IEC 62055-41) produce an AuthenticationResult, a ValidationResult, and a TokenResult;
    c) If STS_ED_AL reports an AuthenticationResult other than “Authentic” then: set DataValue.auth to reflect the AuthenticationResult, and return with status_code = 5 (Authentication failure) and current data_value;
    d) If STS_ED_AL reports a ValidationResult other than “Valid” then: set DataValue.val to reflect the ValidationResult, and return with status_code = 6 (Validation result failure) and current data_value;

e) If STS_ED_AL reports a TokenResult that is neither “Accept” nor provisionally accepted (1stKCT, 2ndKCT, 3rdKCT, 4thKCT) then: set DataValue.token (type TokenResult) to reflect the TokenResult, and return with status_code = 7 (Token execution result failure) and current data_value;

f) Clear the bit in DataValue.tkf that corresponds to this token.

6) Return status_code = 3 (Token execution result OK) and current data_value.

Notes:

- Returning from within an iteration gives effect to the rule to stop at the first token that is rejected (see 7.4.2);
- The status_code indicates which field(s) in DataValue must have meaningful values (see Table 15); the semantics of other fields is not clearly defined (this implementation clears all bits in other fields);
- In spite of the above, DataValue.idrecord should have a meaningful value for any status_code except 4 (Token data type failure);
- DataValue.tkf is optional but should be present if there are multiple elements in RequestMessage.tokens and one of these tokens fails.

6.5 ACTION_response (Meter to POS)

The response shall have the following fields (derived from IEC 62055-41 MeterApplicationProcess):

- AuthenticationResult, a bit string of variable length;
- ValidationResult, a bit string of variable length;
- TokenResult, a bit string of variable length;
- IDRecordResult, a bit string of variable length;

6.6 DLMS/COSEM class_id

The class_id for the STS TokenGateway is defined as 115 (0x0073 Hex) in IEC 62056-6-2.

6.7 OBIS Identification

The OBIS Identification parameters are made up as shown in Table 7.

Table 7 - OBIS Identification

Prepayment objects	class_id	OBIS identification					
		A	B	C	D	E	F
TokenGateway	115 (0x0073)	0	0	19	40…49	0	255

The Utility has the ability to select any of the token gateway OBIS codes in value group D in the range 40 to 49. This value shall be defined in a Companion Specification generated and maintained by the Utility.

6.8 DefinitiveIdentifier

The DefinitiveIdentifier is a unique number allocated to the STSA as a Private Enterprise Number by the IANA (Internet Assigned Numbers Authority). The number allocated to the STSA is 43924 and shall be used for all Request and Response messages in the

DLMS/COSEM protocol layer. Note that the DefinitiveIdentifier is not sent in the message, but only specified in the message definition.

The ASN.1 message is defined in the STSA's private namespace, STSForCosem

{

iso(1)

identified-organization(3)

dod(6)

internet(1)

private(4)

enterprise(1)

stsa(43924)

stsforCosem(1)

}

7 Message – ASN.1 Format

7.1 General

Although A-XDR is used as the encoding rule, the message is formulated using ASN.1 standards.

For a full definition of the ASN.1 BER standard, refer to ITUT –TREC X680 2008-11. Presented in the following clauses is a summary of the data elements used for data encoding.

7.2 Data elements

7.2.1 MessageBlock data elements

Table 8 shows the data elements of the message block for transfer to and from the payment meter.

Table 8 – MessageBlock data elements

Element	Format	Range	Carrier	Ref
TokenDataType	Integer	0-99	Request	7.2.2
IDRecord	Printable string	35 alphanumeric ASCII digits	Request	7.2.3
PRNRecord	Printable string	ASCII string of variable length	Request	7.2.4
TokenData	OCTET STRING	66 bits	Request	7.2.5
AuthenticationResult	Bit string (note1)	variable length	Response	7.2.6
ValidationResult	Bit string (note1)	variable length	Response	7.2.7
TokenResult	Bit string (note1)	variable length	Response	7.2.8
IDRecordResult	Bit string (note1)	Variable length	Response	7.2.3

Note : In an ASN.1 BIT STRING with named bits, bit zero (0) is the leading bit [ITU X.680-0207]. For example in 'AuthenticationResult' the leading bit is 'authentic(0)'. A-XDR encoding of a BIT STRING (that does not have a SIZE constraint) produces a Length field followed by a Content field; the leading bit of the BIT STRING is placed in the left-most bit of the first octet of the Content field [see IEC 61334-6 DLMS UA 1001-2:2000]. For example A-XDR encoding of 'AuthenticationResult' (3 bits) with 'authentic' set and all other bits cleared, will produce the octet sequence { 0x03, 0x80 }. Additional information on bit-string A-XDR encoding may be found in DLMS UA 1001-2.

7.2.2 TokenDataType

The TokenDataType indicates the data format of the transmitted token. This could be, for example a 66 bit token, or a bulk token.

TokenDataType 0 has one or more 66 bit (9 octet) elements in 'tokens' and zero elements in 'longtokens'

The following TokenDataTypes are defined:

Table 9 - TokenDataType

TokenDataType	Token length	TokenDataType value
66 bit TokenData	66 bits	0
Reserved for future use	unspecified	1-99

If the token data type is not recognised by the receiver, the error response “Invalid token data type” must be returned (see 7.4.1).

7.2.3 IDRecord

IEC 62055-41 defines the IDRecord as 35 digits. Transmission of the IDRecord is optional. The POS should include the IDRecord unless it has access to meter configuration data that selects to exclude the IDRecord.

The MeterApplicationProcess must check that IDRecord matches the meter configuration or report a meter configuration mismatch.

This is a status indicator to the physical layer protocol to convey the result of the transmitted IDRecord.

Table 10 - IDRecord result

Result	Value (Hex)	Bit Number	Bit-string value
Meter Configuration OK	0x01	0	‘10000000’ (0x80)
MeterPAN mismatch	0x02	1	‘01000000’ (0x40)
not used (set to zero)	0x04	2	‘00000000’ (0x00)
not used (set to zero)	0x08	3	‘00000000’ (0x00)
EA mismatch	0x10	4	‘00001000’ (0x08)
SGC mismatch	0x20	5	‘00000100’ (0x04)
TI mismatch	0x40	6	‘00000010’ (0x02)
KRN mismatch	0x80	7	‘00000001’ (0x01)
Reserved for future use	0x100 – n	8-n	

7.2.4 PRNRecord

Optional variable length print data intended to be printed at the same time as the coding of the token onto the TokenCarrier, as defined in Table1 of IEC 62055-41

7.2.5 TokenData

This is the 66-bit binary format of the token data as decoded from the TokenCarrier. It is the same data element as is present in the TCDU at the POSToTokenCarrierInterface (see 6.4.2 to 6.4.4 of IEC 62055-41).

7.2.6 AuthenticationResult

This is a status indicator to the physical layer protocol to convey the result from the initial authentication checks.

The format for AuthenticationResult is given in Table 11.

Table 11 - AuthenticationResult values

Result	Value (Hex)	Bit number	Bit-string value	Reference
Authentic	0x01	0	‘10000000’ (0x80)	IEC 62055-41 7.1.2
CRCError	0x02	1	‘01000000’ (0x40)	IEC 62055-41 7.1.2
MfrCodeError	0x04	2	‘00100000’ (0x20)	IEC 62055-41 7.1.2
Reserved for future use	0x08 – n	3 - n		

7.2.7 ValidationResult

This is a status indicator to the Physical Layer Protocol to convey the result from the initial validation checks.

Table 12 - ValidationResult values

Result	Value (Hex)	Bit Number	Bit-string value	Reference
Valid	0x01	0	‘10000000’ (0x80)	IEC 62055-41 7.1.3
OldError	0x02	1	‘01000000’ (0x40)	IEC 62055-41 7.1.3
UsedError	0x04	2	‘00100000’ (0x20)	IEC 62055-41 7.1.3
KeyExpiredError	0x08	3	‘00010000’ (0x10)	IEC 62055-41 7.1.3
DDTKError	0x10	4	‘00001000’ (0x08)	IEC 62055-41 7.1.3
Reserved for future use	0x20 – n	5-n		

7.2.8 TokenResult

This is a status indicator from the MeterApplicationProcess to convey the result after processing the Token so that the Physical Layer Protocol can take the appropriate action. See also 7.1.4 of IEC 62055-41 of for definition of the TokenResult values.

Table 13 TokenResult values

Result	Value (Hex)	Bit Number	Bit-string value	Reference
Accept	0x01	0	‘10000000000000000’ 0x8000	IEC 62055-41 7.1.5
1stKCT (Note2)	0x02	1	‘0100000000000000’ (0x4000)	IEC 62055-41 7.1.5
2ndKCT (Note2)	0x04	2	‘0010000000000000’ (0x2000)	IEC 62055-41 7.1.5

3rdKCT (Note2)	0x08	3	'0001000000000000' (0x1000)	IEC 62055-41 7.1.5
4thKCT (Note2)	0x10	4	'0000100000000000' (0x0800)	IEC 62055-41 7.1.5
OverflowError	0x20	5	'0000010000000000' (0x0400)	IEC 62055-41 7.1.5
KeyTypeError	0x40	6	'0000001000000000' (0x0200)	IEC 62055-41 7.1.5
FormatError	0x80	7	'0000000100000000' (0x0100)	IEC 62055-41 7.1.5
RangeError	0x0100	8	'0000000010000000' (0x0080)	IEC 62055-41 7.1.5
FunctionError	0x0200	9	'0000000001000000' (0x0040)	IEC 62055-41 7.1.5
InvalidTokenDataType (Note1)	0x0400	10	'0000000000100000' 0x0020	
Reserved for future use	0x0800 – n	11-n		

Note 1: an InvalidTokenDataType error will occur if the server does not support the token type sent. For example, if the server does not support a bulk token (which has more than the normal 66 bits).

Note 2: in a key-change operation, each token in the set shall return the token number (1stKCT, 2ndKCT, 3rdKCT, 4thKCT, except the last token in the key-change operation which shall return 'Accept' if the set is accepted. As an example: in a two key-change set, if the first token entered is KCT1, the server shall return '1stKCT', and on acceptance of the second token, the server shall return 'Accept', and not '2ndKCT'.

If the order of the tokens is the 2nd KCT followed by the 1st KCT, then the server shall return '2ndKCT' after acceptance of the first token, and 'Accept' after acceptance of the second token. See also 8.2 of IEC 62055-41 for acceptance of tokens.

7.3 Message definitions

7.3.1 General

The various messages supported are given below.

7.3.2 RequestMessage

Each RequestMessage comprises the fields as shown in Table 14.

Table 14 – RequestMessage field definitions

Field	Size/Value	Type	Optional (Yes/No)
TokenDataType	0-99	Integer	No
IDRecord	35 Digits	Printable String	Yes
PRNRecord	variable length	Printable String	Yes
TokenData	66 bits	Octet Strings	No

Note: the Token data bit string must be left padded with 6 zeros to make up a full set of octets.

7.3.3 RequestMessage schema

The TCDU is encoded into an ASN1 RequestMessage with A-XDR encoding.

StsForCosem {iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) stsa(43924) stsForCosem(1)} DEFINITIONS AUTOMATIC TAGS ::= BEGIN

```
RequestMessage ::= SEQUENCE
{
        tokendatatype Integer (0..99),
```

```
        idrecord        PrintableString (SIZE(35)) OPTIONAL,
        prnrecord       PrintableString, OPTIONAL,
        tokens          SEQUENCE OF OCTET STRING(Size(9)),

        longtokens      SEQUENCE OF OCTET STRING OPTIONAL
        …
}
END
```

These data fields make up the Contents portion of the ASN1 defined message as shown in the worked example in Annex A.2.

7.4 ResponseMessage

7.4.1 ResponseMessage schema

The TCDU shall be encoded into an ASN1 response message with A-XDR encoding in accordance with 5.4 and Clause 7 of IEC 62055-41.

The token status response is defined in 7.4.2 as follows:

```
token_status ::= structure
{
        status_code:            enum
        data_value:             bit-string (refers to the DataValue)

}
```

Where the status_code is an enumerated value (see 7.4.2), and the data_value is constructed by ASN.1 encoding DataValue to produce a bit-string.

```
StsForCosem {iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) stsa(43924) stsForCosem(1)} DEFINITIONS AUTOMATIC TAGS ::= BEGIN

DataValue ::= SEQUENCE
{
        auth AuthenticationResult,
        val ValidationResult,
        token TokenResult,
        idRecord IDRecordResult, (set to valid if not included in RequestMessage)
        tkf TokenFailure OPTIONAL
        …
}
AuthenticationResult ::= BIT STRING
{
        authentic (0),
        crcError(1),
        mfrCodeError(2)
        -- other bits RFU
}
ValidationResult ::= BIT STRING
{
        valid(0),
        oldError(1),
        usedError(2),
        keyExpiredError(3),
        dDTKError(4)
        -- other bits RFU
```

```
}
TokenResult ::= BIT STRING
{
        accept(0),
        kCT1(1),
        kCT2(2),
        kCT3(3),
        kCT4(4),
        overflowError(5),
        keyTypeError(6),
        formatError(7),
        rangeError(8),
        functionError(9),
        invalidTokenDataType(10),
        -- other bits RFU
}
IDRecordResult ::= BIT STRING
{
        valid(0),
        meterPanMismatch(1),
        NotUsed2(2) set to zero,
        NotUsed3(3) set to zero,
        eaMismatch(4),
        sgcMismatch(5),
        tiMismatch(6),
        krnMismatch(7),
        -- other bits RFU
}
TokenFailure ::= BIT STRING
{
        token1Failure(0),
        token2Failure(1),
        token3Failure(2),
        token4Failure(3),
        -- other bits RFU
}
END
```

7.4.2 DLMS/COSEM token_status

The following structure is defined in IEC 62056-6-2 for the token status returned after a RequestMessage:

```
token_status ::= structure
{
        status_code:        enum
        data_value:         bit-string (refers to the DataValue)
}
```

status_code::= enum

(0) Token format result OK

(1) Authentication result OK

(2) Validation result OK

(3) Token execution result OK

(4) Token data type failure (refers to the ASN1 message)

(5) Authentication failure

(6) Validation result failure

(7) Token execution result failure

(8) Token Received but not yet processed

Table 15 shows the DLMS/COSEM token_status enumerated fields, together with the matching IEC62055-41 token status conditions.

Table 15 – DLMS/COSEM token_status result

DLMS/COSEM enumerated value	STS result
(0) Token format result OK	Not used
(1) Authentication result OK	Not used
(2) Validation result OK	Not used
(3) Token execution result OK	Token accepted
(4) Token data type failure	Set if unsupported data type or invalid token data format - returned if the DLMS/COSEM token (i.e. ASN.1-encoded RequestMessage) cannot be decoded or understood successfully
(5) Authentication failure	Set if authentication fails – see AuthenticationResult bit-string for result. DataValue (type TokenFailure), if present, indicates which token (from RequestMessage.tokens) failed
(6) Validation result failure	Set if validation fails - see ValidationResult bitstring for result. DataValue (type TokenFailure), if present, indicates which token (from RequestMessage.tokens) failed
(7) Token execution result failure	Set if token processing fails – see TokenResult bit-string for result. DataValue (type TokenFailure), if present, indicates which token (from RequestMessage.tokens) failed
(8) Token received but not yet processed	Set if the server is not able to process the token type that it has received (i.e. bulk token which is not the standard 66 bit format)

Note that the values shown in the DLMS/COSEM enumerated values column are mutually exclusive: only one value can be returned on a token status response message.

The value to be used in the response message will be the first failure that is encountered during the processing of the token in the MeterApplicationLayer or MeterApplicationProcess. Processing of tokens shall stop after detection of the first failure.

The enumerated values (0), (1), and (2) are redundant since if a token is decrypted correctly, enumerated value (3) will be returned.

Enumerated value (4) format failure could be due to an unsupported data type, or an invalid token data format.

If authentication, validation or execution failures occur, the failure result will be contained in the corresponding bit-string value.

Enumerated value (7) is set only if the error encountered is one of the failures specified in 7.4.1 in the TokenResult bit-string (bits 4,5,6,7, or 8).

Annex A
(informative)

Worked examples

A.1 Introduction

This annex shows worked examples of the messages described in this standard.

A.2 Worked example - RequestMessage

A.2.1 RequestMessage

For this example the data shown in Table 16 are used.

Table 16 – RequestMessage data

Element	Description
TCT	23 (arbitrarily chosen)
ID Record	60072700000000000900000207123456011
PRNRecord	“Test Message”
SGC	123456
TI	01
KRN	1
KEN	FF
Transfer Amount	0.1kWh
Issue Date/Time	2004-03-01 13:55
Vending Key	ABABABABABABABAB
EA	EA07
DKGA	02
Numeric token	1252 9416 9400 6201 3255
Binary Token	00101011011**11100001**01101110**01010010**11000011**00101000**01110111**01000111**
Hex Token	00ADE16E52C3287747

A.2.2 Values for RequestMessage worked example

```
test RequestMessage ::=
{
        tokendatatype 0,
        idrecord "600727000000000009===0207123456011",
        prnrecord “Test Message”
        token{ '00ADE16E52C3287747'H}
}
```

A.2.3 Content (C)

The content is made up as shown in Table 17.

Table 17 – MessageBlock Content

Name	Content
TokenDataType	00
IDRecord	36 30 30 37 32 37 30 30 30 30 30 30 30 30 30 30 39 30 30 30 30 30 32 30 37 31 32 33 34 35 36 30 31 31 Hex
PRNRecord	54 65 73 74 20 4D 65 73 73 61 67 65
TokenData	00 AD E1 6E 52 C3 28 77 47 Hex (left padded with 6 zeros)

A.2.4 A-XDR encoded message

In this clause we will take the ASN.1 message and encode it using the A-XDR encoding rules. For this form of encoding, Table 18 shows the general structure of the encoded message.

Table 18 – General structure of encoded message

Identifier	Length	Contents

Length values are only required where the contents are not of a fixed length.

The Contents element is shown in Table 19.

Note that since lengths for all contents components are known, there is no need to encode length values into the message payload before each contents component, except for the Tokens contents to cater for multiple tokens, and the PRNRecord.

Similarly, an Identifier is not required since both transmitter and receiver both know the message specification.

Recall the RequestMessage schema given in 7.3.3:

```
RequestMessage ::= SEQUENCE
{
        tokenDataType Integer (0..99),
        idRecord        PrintableString (Size(35)) OPTIONAL,
        prnRecord       PrintableString OPTIONAL,
        tokens          SEQUENCE OF OCTET STRING (Size(9))
        longtokens      SEQUENCE OF OCTET STRING OPTIONAL
        …
}
END
```

The A-XDR encoded message is shown in Table 19

Table 19 – A-XDR encoded RequestMessage

TokenDataType	Usage Flag (UF1)	IDRecord	Usage Flag UF2	L1	PRN Record	L2	Tokens	Usage Flag UF3
00 (Hex)	01 (Hex)	36 30 30 37 32 37 30 30 30 30 30 30 30 30 30 30 30 39 30 30 30 30 30 32 30 37 31 32 33 34 35 36 30 31 31 (Hex)	01 (Hex)	0C (Hex)	54 65 73 74 20 4D 65 73 73 61 67 65 (Hex)	01 (Hex)	00 AD E1 6E 52 C3 28 77 47 (Hex)	00 (Hex)

A non-zero value for Usage Flag UF1 indicates that the IDRecord is present in the message. A zero value for Usage Flag UF1 indicates that there is no IDRecord field present in the message. Similarly for UF2 and PRN Record, UF3, and longtokens.

Note that the length value for the entire message is not required since the number of components of the SEQUENCE are known and defined in the ASN.1 definition of the message.

The length value (L1) is required since the PRNRecord component is of variable length. In this example it has a value of 12 (0x0C) since the PRNRecord is 12 bytes long.

The length value (L2) is required since this SEQUENCE OF component is of variable length. In this example it has a value of 1 since only one token is sent. The number of octets in the token itself (9) is specified in the message definition.

A.2.5 DLMS/COSEM encoded message

Using A.2.4 and the DLMS/COSEM definitions given in IEC 62056-6-2, and the various DLMS/COSEM references given in Clause 2, the encoded DLMS/COSEM message becomes as shown in Table 20.

Table 20 – DLMS/COSEM encoded RequestMessage

DLMS/COSEM RequestMessage									
C3	01	C1	0073	0000132800FF	01	01	09	3E	[00][01]01[ID Record]01 0C[L1][PRNRecord] [L2] [Tokens]00
a	b	c	d	e	f	g	h	i	j

Note: all values in line 1 of Table 20 are in Hex.

Each item in Table 20 is shown in Table 21.

Table 21 – Items of the DLMS/COSEM encoded RequestMessage

Item	Description
a	APDU tag for ActionRequest
b	Action-Request-Normal
c	InvokeIdAndPriority Value - 0xC1 (Binary 1 100 0001) => High Priorty Confirmed Request InvokeId Value = 1
d	ClassId Value = 0x0073 = 115
e	InstanceId Value = 0x0000132800FF = 0-0:19.40.0.255
f	MethodId Value

g	Optional Data is present - The MethodInvocationParameters consists of optional data which may (value is non-zero) or may not (value is zero) be present
h	Data type – (09) = OCTET STRING
i	Octet String Length – 0x3E = 62 bytes
j	Message payload octet string. TokenDataType, IDRecord, PRNRecord, L1, L2, UF1, UF2, UF3, and Tokens as per Table 19 values

A.3 Worked example – ResponseMessage

A.3.1 DataValue

For the response message, we assume the response in A.3.2 to the request message given in the worked example in A.2, and using the ResponseMessage schema defined in 7.4.1.

Note that the SEQUENCE size (number of entries) is defined as 5, so this value does not need to be encoded. The length of the BIT STRING values is not defined in the ASN.1 specification message above (since this can be expanded at a later stage), therefore the length of each BIT STRING must be encoded in the message.

A.3.2 Test values for DataValue

For the purposes of this example, we use the following test values:

```
test DataValue ::=
{
        auth ‘10000000’B,
        val ‘10000000’B,
        token ‘10000000000’B,
        idRecord ‘10000000’B,
        tkf ‘10000000’B
}
```

A.3.3 A-XDR encoded DataValue

The A-XDR encoded DataValue is shown in Table 22.

Table 22 – A-XDR DataValue

L1	Auth Result	L2	Validation Result	L3	Token Result	L4	IDRecord Result	UF1	L5	Token Failure
0x03	0x80	0x05	0x80	0x0B	0x8000	0x08	0x80	0x01	0x04	0x80

Where L1, L2, L3, L4 and L5 represent the size of the BIT STRINGS AuthenticationResult, ValidationResult, TokenResult, IDRecordResult, and TokenFailure respectively. Note that the Token Failure bit-string is optional so UF1 is always present. If there is no token failure, then L5 and the token failure value are not present.

A.3.4 DLMS/COSEM encoded ResponseMessage

According to DLMS/COSEM rules then, the entire message is as shown in Table 23.

Table 23 – DLMS/COSEM encoded ResponseMessage

DLMS/COSEM ResponseMessage												
C7	01	C1	00	01	00	02	02	16	05	04	60	Table 22 data
a	b	c	d	e	f	g	h	i	j	k	l	m

The DLMS/COSEM encoded ResponseMessage (token_status) items in Table 23 are shown in Table 24.

Table 24 – DLMS/COSEM encoded ResponseMessage items

Item	Description
a	APDU tag for ActionResponse
b	Action-Response-Normal
c	InvokeIdAndPriority Value - 0xC1 (Binary 1 100 0001) => High Priorty Confirmed Request InvokeId Value = 1
d	ResultValue = Success
e	Usage Flag = Present
f	Get Data Result Choice = “data”
g	Data Choice = “structure”
h	Structure size = 2
i	Enum data type = 22 (“16” in Hex)
j	Enum Value = “Authentication failure”
k	Bit-string data type = 4 (“04” in Hex)
l	BitString length
m	message payload

Bibliography

IEC 62055-21, *ELECTRICITY METERING – PAYMENT SYSTEMS Part 21: Framework for standardization*

CMU/SEI-93-TR-10 ESC-TR-93-187, *The Use of ASN.1 and XDR for Data Representation in Real-Time Distributed Systems: B. Craig Meyers, Gary Chastek - October 1993*

DLMS UA 1001-2;2000, *COSEM A-XDR encoding rule*