This document is based on
German draft SML version 1.04
dated 13.07.2010
including some minor corrections

Editorial Version by MW, dated 07.01.2011
Update during WG-Meeting 17.01.2011
Editorial Version by MW, dated 24.03.2011
Update during WG-Meeting 05.05.2011 / restructured according to meeting decisions
Update during WG-Meeting 02.12.2011

SML Container Services

The first page has to be adapted later.

List of contents

List of illustrations

List of tables

List of abbreviations

Units:

In regard to physical measured variables and units, the agreements laid down in SI (see ISO 1000) shall apply.

Relevant abbreviations:

The abbreviations below can be followed by Arabic numerals enabling distinctions to be made between multiple variants of the same function/signal.

ASN.1	⇔ Abstract Syntax Notation One,
BER	⇔ Basic Encoding Rules,
CCITT	⇔ Comité Consultatif International Télégraphique et Téléphonique,
CR	⇔ Carriage Return,
DLMS	⇔ Device Language Message Specification,
EN	⇔ European Standard,
ID	⇔ Identification Number,
IEC	⇔ International Electrotechnical Commission,
IEEE	⇔ Institute of Electrical and Electronics Engineers,
IP	⇔ Internet Protocol,
ISO	⇔ International Organisation for Standardisation,
LAN	⇔ Local Area Network,
LSB	⇔ Least Significant Bit, niederwertigstes Bit,
MDE	⇔ Mobile data acquisition device,
MSB	⇔ Most Significant Bit, höchstwertigstes Bit,

OBIS	⇔	Object Identification System,
OSI	⇔	Open Systems Interconnection Reference Model,
RS232	⇔	Serial interface,
S-I	⇔	Seconds Index,
SML	⇔	Smart Message Language,
TAG	⇔	Feature / identification / tagging for the coding of data elements,
TCP	⇔	Transmission Control Protocol,
TL	⇔	Type-Length,
UDP	⇔	User Datagram Protocol,
WAN	⇔	Wide Area Network,
XML	⇔	Extensible Markup Language.

SPECIFICATION OF THE SMART MESSAGE LANGUAGE SML

1 Scope

The goal of the specification was to find a maximally simple structure also suitable for implementation in low-power embedded systems that can be used for data procurement over wide-area routes. This specification defines a communication protocol for applications in the environment of data procurement and equipment parameterisation.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

CCITT-CRC16, *Standard of the CCITT for checksum calculation*

IEC 62056-21, *Measuring electrical energy – meter reading transmission, Part 21: Direct local data exchange*

IEC 62056-46, *Measuring electrical energy - meter reading transmission, tariff and load control – Part 46: Use of the HDLC protocol in the connection layer*

IEC 62056-61, *Measuring electrical energy – meter reading transmission, Part 61: OBIS Object Identification System*

IEC 62056-62, *Measuring electrical energy - meter reading transmission, tariff and load control – Part 62: Interface Classes*

ISO 1000, *SI units and recommendations for the use of their multiples and of certain other unitsSI units and recommendations for the use of their multiples and of certain other units*

ISO 8859-15, *Information technology – 8-bit-indiviudal-byte-coded character sets – Part 15: Latin Alphabet No. 9*

3 Terms and definitions

Some important terms used in this document are explained or defined below.

3.1 SML file

denotes an information unit that, completely detached from the transport technology involved (internet, telephone, ...), is self-contained. SML files can in this sense be comprehended as self-contained information units that (just like an email) are embedded and transmitted in a protocol (see Fig. 2).The approach of using SML files means the concept is independent of the task of having to define specific protocols for information interchange. Instead, all that is demanded is to select a certain protocol (e.g. HTTP, FTP, ...) in a particular duty case and parameterise this appropriately. If SML files are used as files on computer systems, these files must be noted without the use of additional frames and utilising the coding defined with Section 7, unless the particular application involved explicitly specifies a divergent stipulation.

4 Basic structure of SML

The basic structure contains the following elements:

– Smart Message Language defines a file structure / document structure for recording the useful loads between the end points

– SML Binary Encoding defines a packed binary coding for SML;

– SML XML Encoding defines the coding of SML in XML;

– SML Transport Protocol, required for serial point-to-point links.

SML messages, see no. 5, can, like an email in the final analysis, be transported over stateless, secure communication paths. For the duty scenario being targeted, the following model can accordingly be adduced for an overview:

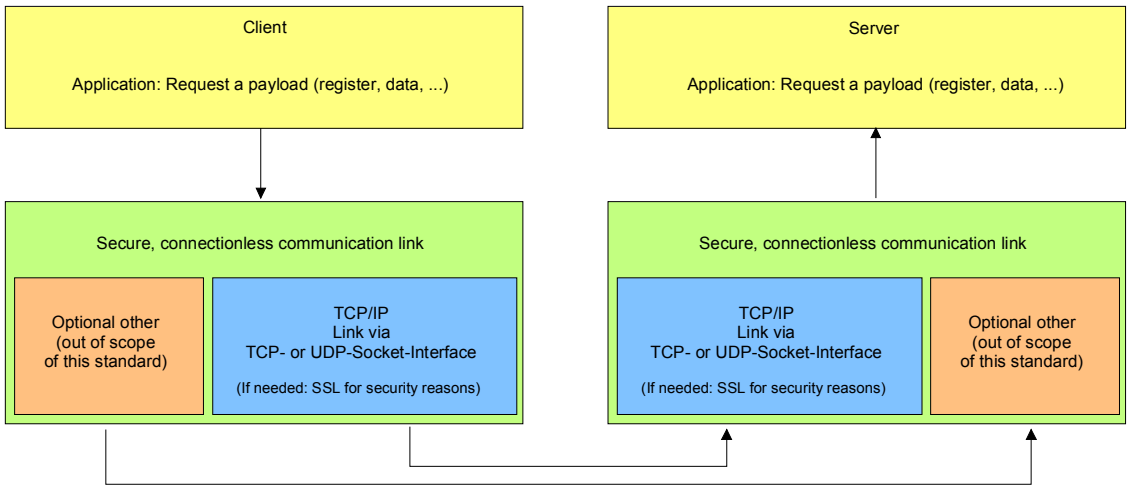

Fig. 1: SML messages and communication paths

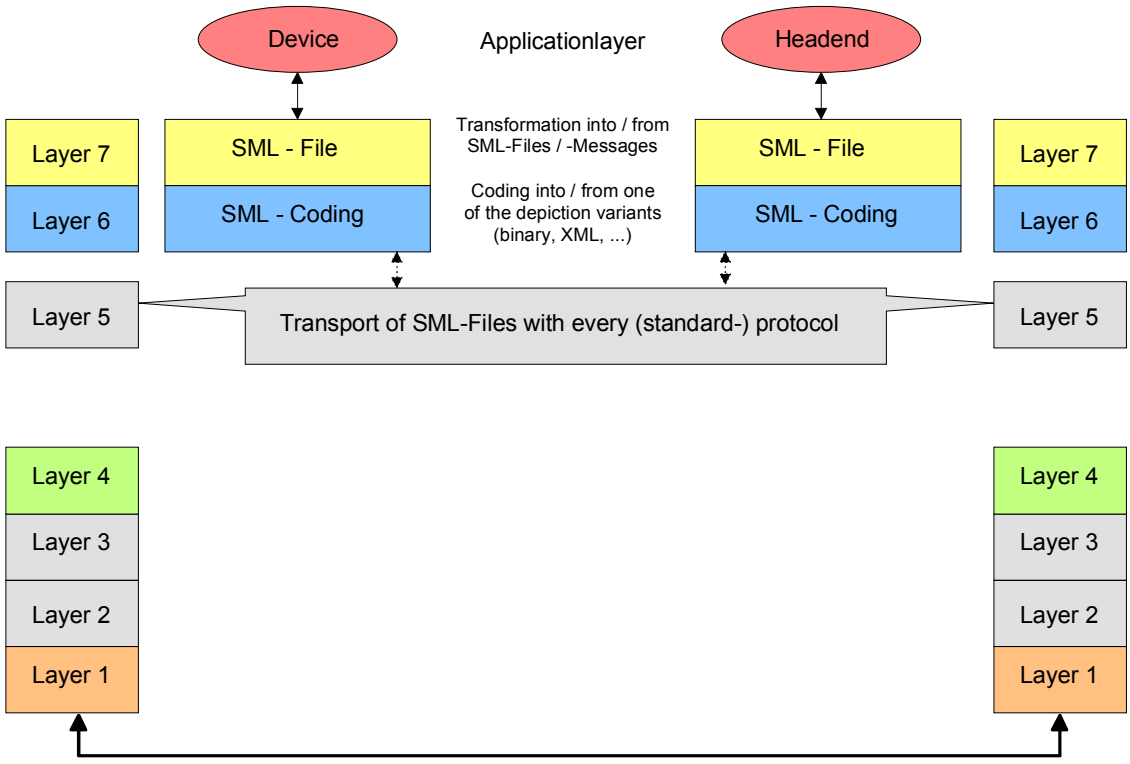

Fig. 2: SML communication model.

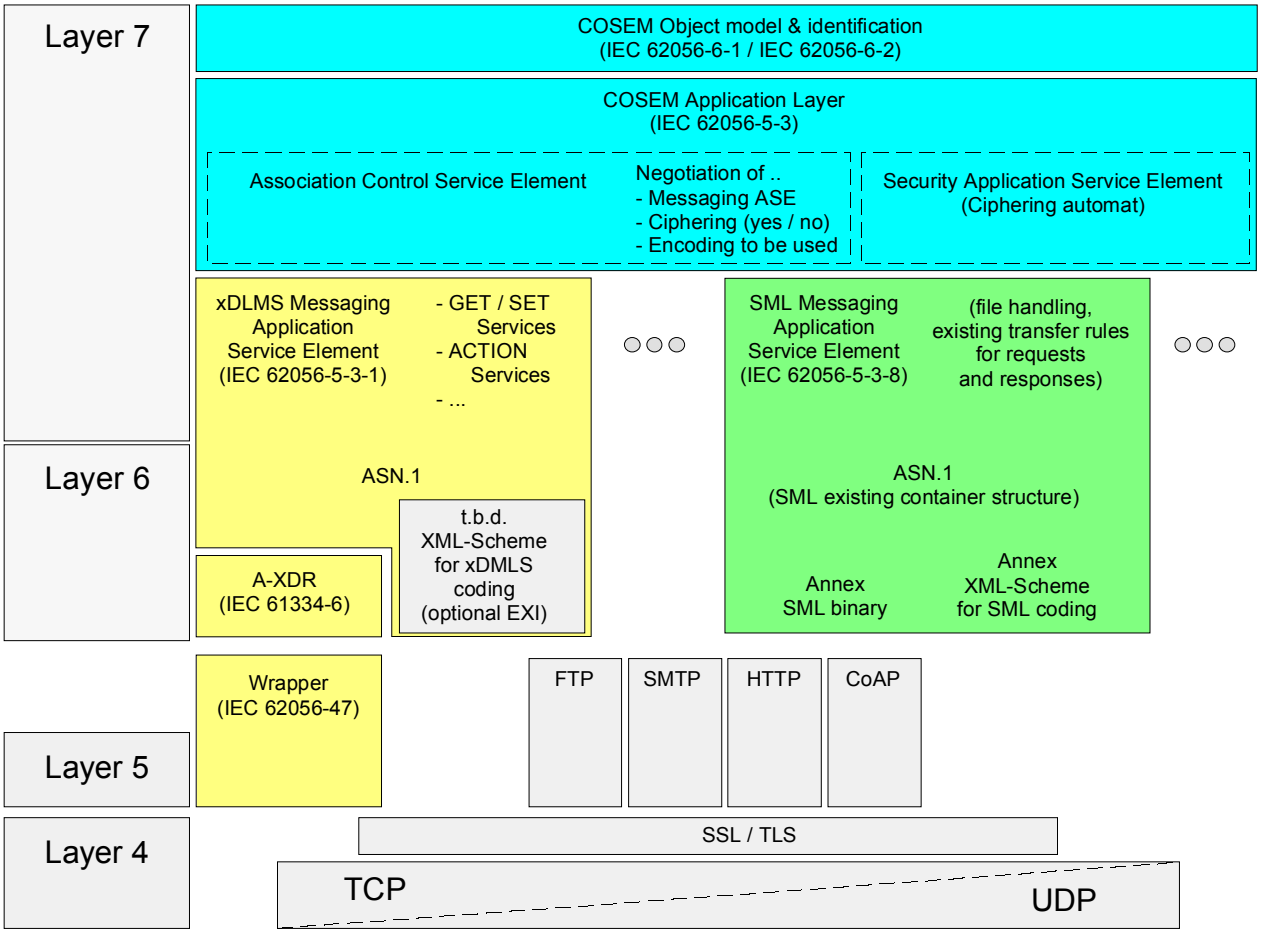

Fig. 3: Location of SML within the field of European and international standards

5 Basic Structure of Smart Message Language SML

An SML file is always structured as a chain of SML messages.

SML files can be segmented in order to reduce the file size involved.

SML files can occur in the following variants:
... SML order file,
... SML response file or
... SML combi-file

SML order files contain the orders (“Requests”). SML response files group together the responses to the requests.

Each SML order file begins with exactly one and contains exactly one SML_...Open.Req message. It ends with exactly one and contains exactly one SML_...Close.Req message. This specification also applies for SML response files to which no SML order file can be assigned, where the ‘response’ variant must be used instead of the ‘request’ variant.

SML response files to which SML order files can be assigned begin with exactly one SML_...Open.Res- or one SML_Attention.Res message. They end with exactly one SML_...Close.Res- or one SML_Attention.Res message

For the use of SML per transport media with lower performance[1] SML messages can also be sent without the SML file frame and exclusively as „Response without Request“. This use case must be defined explicitly by the application.

An SML combi-file contains the SML messages of an SML order file plus the SML messages of the associated SML response file(s).

6 SML messages

An SML message is either a ‘Request message’ or a ‘Response message’.

An SML message comprises task and assigned attributes.

```
(A)   SML_Message          ::=   SEQUENCE
      {
        transactionId              Octet String,
        groupNo                    Unsigned8,
        abortOnError               Unsigned8,
        messageBody                SML_MessageBody,
        crc16                      Unsigned16,
        endOfSmlMsg                EndOfSmlMsg
      }
```

The ‘transactionId’ is formed by the client in an unambiguous form[2] at generation of ‘request messages’. Each ‘response message’ reflects back the unchanged transaction number belonging to its ‘request’, so that a ‘response’ can always be assigned to the associated ‘request’.

[Reference Ref. 1, see below]
When an SML file is created in the sense of ‘push mode’, and by reason of the principle involved no SML request messages exist for its SML response messages, the creator of these SML response messages automatically generates unambiguous transaction numbers.

The ‘groupNo’ attribute permits the formation of SML message groups. This mechanism shall be used for stating which SML messages must be processed in a particular sequence, and which may be executed on the side (in the sense of simultaneous work of both the first and second kind).

As a general principle, the groups are processed sequentially, in the same order as they occur within an SML file.

Mixing SML messages from different groups is not allowed.

Messages inside a particular group can be processed by the recipient either serially or on the side.

The checksum (element ‘crc16’) must be calculated as CRC16 in accordance with IEC 62056-46. Calculation begins with the first byte for ‘SML_Message’ and ends with the last byte for

---

[1] In this specification only the variants ‘short range radio’ and ‘PLC connections’ between sensors and actors are considered as *transport media with lower performance*.

[2] The unambiguity must in each case be assured within the context of the generator of the Transaction Number; a worldwide unambiguity is not demanded (comparable to the MAC address with Ethernet). Thanks to the other particulars in the SML-Open message, the unambiguity can be assigned to the generator involved.

‘messageBody’. This means the bytes of elements ‘crc16’ and ‘zero’ are excluded from checksum calculation.

When an SML request message is received that could be decoded and whose checksum is faulty, this SML request message must be answered with an ‘SML_Attention’ (81 81 C7 C7 FE 0B, see Tab. 2) and the field ‘transactionId’ according to reference Ref. 1. If in this case the ‘SML_CloseRequest’ message is received, the fault must be ignored and in the response a correct ‘SML_Close-Request’ must be sent whose ‘transactionId’ field is set according to reference Ref. 1.

If the ‘SML_OpenRequest’ message is received with a faulty checksum or a faulty ‚SML_PublicOpen.Req’ structure or a missing clientId or a missing reqFileId, the entire SML file must be ignored[3].

If a SML-Request-file was received which could be decoded but did not start with ‚SML_Open-Request’ then the answer has to be a SML-Request file with one SML Attention message and error code ‘Unexpected SML message’. A further processing and/or a correct answering to the subsequent SML messages is forbidden.

If an SML message is received that cannot be decoded, the following procedure must be adopted:

- If the SML message concerned is the first message, the entire SML file will be rejected (this applies both for SML request files and also for SML response files).
- If one of the following SML messages is involved, then as the response for the SML message concerned an SML-Attention (81 81 C7 C7 FE 08, see Tab. 2) is transmitted followed by an SML-Close (applies only for SML order files). All further SML messages of the SML file involved will be ignored (in the case of both SML order files and SML response files).
- The transactionId’ for the SML_Attention’ as well as for the following SML_Close must be generated according to reference Ref. 1.
- If a SML_Close request could be recognised but the CRC inside is wrong the answer has to be a SML_Close response with a transactionId according to reference Ref. 1.

SML messages generally have to be aggregated to SML files.

Tab. 1: Example for using the ‘Group Number’ feature

Sequence of SML messages	Trans-action Number	Group-No.	SML message	Comment
0	0	1	OPEN	Is executed first.
1	1	4	GET_ProfilPack	First group, is executed as second block. For example, read out three load profiles.
2	2	4	GET_ProfilPack	
3	3	4	GET_ProfilPack	
4	4	7	GET_ProfilPack	Second group, is executed as third action, For example, read out a load profile, the billing list and the logbook.
5	5	7	GET_ProfilList	
6	6	7	GET_ProfilList	
7	7	8	SET_ProcParameter	Is executed as fourth action. For example, correct the reference time

[3] Ignoring means: The received content must be rejected. No answer is generated.

Sequence of SML messages	Trans-action Number	Group-No.	SML message	Comment
8	8	8	CLOSE.Req	Is processed last.

The Transaction Number is thus merely the feature for assigning the responses to the associated requests. It has no influence whatever on the sequence for executing the SML messages at the recipient.

The 'abortOnError' attribute specifies what procedure is to be followed in the event of errors in executing the SML message, with the concluding Close always having to be executed and non-executed requests always generating a response with the 'not executed' error message as the feedback message:

(B) abortOnError ⇔ 0x00 Continue execution,
abortOnError ⇔ 0x01 Continue execution
as from next group,
abortOnError ⇔ 0x02 Continue execution beginning
at the ongoing group, then
do not execute any
more groups.
If a '0x02' can be found
within the same group
followed by a '0x01',
execution must be aborted
immediately.
abortOnError ⇔ 0xFF Abort execution immediately.

```
(C)   SML_MessageBody          ::=   CHOICE
      {
        OpenRequest                  [0x00000100]   SML_PublicOpen.Req
        OpenResponse                 [0x00000101]   SML_PublicOpen.Res

        CloseRequest                 [0x00000200]   SML_PublicClose.Req
        CloseResponse                [0x00000201]   SML_PublicClose.Res

        GetProfilePackRequest        [0x00000300]   SML_GetProfilePack.Req
        GetProfilePackResponse       [0x00000301]   SML_GetProfilePack.Res

        GetProfileListRequest        [0x00000400]   SML_GetProfileList.Req
        GetProfileListResponse       [0x00000401]   SML_GetProfileList.Res

        GetProcParameterRequest      [0x00000500]   SML_GetProcParameter.Req
        GetProcParameterResponse     [0x00000501]   SML_GetProcParameter.Res

        SetProcParameterRequest      [0x00000600]   SML_SetProcParameter.Req

        GetListRequest               [0x00000700]   SML_GetList.Req
        GetListResponse              [0x00000701]   SML_GetList.Res

        GetCosemRequest              [0x00000800]   SML_GetCosem.Req
        GetCosemResponse             [0x00000801]   SML_GetCosem.Res

        SetCosemRequest              [0x00000900]   SML_SetCosem.Req
        SetCosemResponse             [0x00000901]   SML_SetCosem.Res

        ActionCosemRequest           [0x00000A00]   SML_ActionCosem.Req
        ActionCosemResponse          [0x00000A01]   SML_ActionCosem.Res
        AttentionResponse            [0x0000FF01]   SML_Attention.Res
      }
```

6.1 SML Messages

6.1.1 SML_PublicOpen.Req

The SML_PublicOpen.Req must always be present at the beginning of an SML order file. It serves to identify the client and the authentication per user/password and the assignment of SML response file(s) to the SML order file.

Each client has to respond to the SML_PublicOpen.Req either with an SML_PublicOpen.Res or an SML_Attention.Res. The empty ‘ServerID’ must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(D)   SML_PublicOpen.Req      ::=   SEQUENCE
      {
        codepage                    Octet String OPTIONAL,
        clientId                    Octet String
        reqFileId                   Octet String
        serverId                    Octet String OPTIONAL,
        username                    Octet String OPTIONAL,
        password                    Octet String OPTIONAL,
        smlVersion                  Unsigned8 OPTIONAL,
      }
```

Per ‘codepage’, if the value is stated, a codepage other than the default codepage is agreed. The ‘Codepage’ specifies which character set is to be used for interpreting strings. If the value is not stated, ‘ISO 8859-15’ will be used.

The content for ‘Codepage’ itself must always be noted as a string in ISO 8859-15.

The ‘serverId’ parameter, if stated, enables to address an SML data source (a specific measuring device) or a software module selectively per ‘PublicOpen.Req’ in the sense of an address. In this the case, all subsequent SML messages must likewise state the parameter ‘serverId’ with content. If the parameter ‘serverId’ is absent in one of the subsequent messages, the response for the SML message concerned must be an ‘SML_Attention’ (81 81 C7 C7 FE 0C, see Tab. 2).

If the ‘serverId’ parameter is absent, the request will be evaluated as a ‘broadcast’. In this case, the ‘serverId’ parameter must likewise be absent in all subsequent SML messages. SML messages that in this situation nonetheless state the ‘serverId’ parameter must be ignored. In all cases to the final ‚SML_Close’ must be answered.

The ‘clientId’ parameter is generated by the client at generation of an SML order file and serves for unambiguous addressing of the client’s SML response file.

The ‘reqFileId’ parameter designates in a form unambiguous throughout the system[4] a specific SML order file/SML response file tuple. It enables SML responses to be assigned to their requests.

The ‘reqFileId’ parameter supplies unambiguous identification for the SML file, e.g. formed from the ongoing timestamp.

If the ‘sml-Version’ parameter is absent, Version 1 will be assumed as standard.

6.1.2 SML_PublicOpen.Res

SML_PublicOpen.Res always stands at the beginning of an SML response file. It serves to identify the SML response file for the associated SML order file.

---

[4] The unambiguity must in each case be assured within the context of the generator of ‘reqFileId’; a worldwide unambiguity (comparable to the MAC address with Ethernet) is not demanded. Thanks to the other particulars in the SML-Open message, the unambiguity can be assigned to the generator involved.

```
(E)   SML_PublicOpen.Res      ::=   SEQUENCE
      {
        codepage                    Octet String OPTIONAL,
        clientId                    Octet String OPTIONAL,
        reqFileId                   Octet String,
        serverId                    Octet String,
        refTime                     SML_Time OPTIONAL,
        smlVersion                  Unsigned8 OPTIONAL,
      }
```

The ‘SML_Time’ element is stated either as a seconds index or as a timestamp.

```
(F)   SML_Time                ::=   CHOICE
      {
        secIndex                    [0x01]   Unsigned32,
        timestamp                   [0x02]   SML_Timestamp
        localTimestamp              [0x03]   SML_TimestampLocal
      }
```

If a timestamp is involved, this is always formed in seconds, starting from 01.01.1970, 00:00:00 (UNIX reference time).

```
(G)   SML_Timestamp           ::=   Unsigned32
```

If a local time stamp is involved it has to be given as follows:

```
(H)   SML_TimestampLocal      ::=   SEQUENCE
      {
        timestamp                   SML_Timestamp,
        localOffset                 Integer16,
        seasonTimeOffset            Integer16
      }
```

The element ‚localOffset’ must be given in minutes.

The element ‚seasonTimeOffset’ must be given in minutes.

The local time results as follows:

localTime = Timestamp + localOffset + seasonTimeOffset

If a time value is invalid, it must be coded by ‘0xFFFFFFFF’. If in that case it is a SML_TimestampLocal, the values for ‘localOffset’ and ‘seasonTimeOffset’ must be coded by ‘0x0000’.

If the response is supplied by a device or a software module that does not itself possess a piece of time information, this particular will be absent in ‘PublicOpen.Res’.

If the ‘smlVersion’ parameter is absent, Version 1 will be assumed as standard.

If the SML file concerned is an SML response file to which an SML request file belongs, the ‘reqFileId’ element of the SML request file will be returned.

If the SML file is an SML response file to which no SML request file belongs, the ‘reqFileId’ element, comparable to generating this attribute when creating an SML request file, must be set to a name unambiguous throughout the system.

If the SML file is an SML response file to which no SML request file belongs, the ‘clientId’ element may be omitted; in all other cases, the value for ‘client-ID’ supplied with ‘PublicOpen.Req’ must be entered there.

The ‘refTime’ field supplies the reference time for creating the SML response file.

6.1.3 SML_PublicClose.Req

The SML_PublicClose.Req must always be present at the end of an SML order file. It terminates this file.

Each client must respond to the SML_PublicClose.Req with an SML_PublicClose.Res or an SML_Attention.Res. The empty ‘ServerID’ must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(I)   SML_PublicClose.Req      ::=    SEQUENCE
      {
        globalSignature                SML_Signature OPTIONAL
      }
```

6.1.4 SML_PublicClose.Res

SML_PublicClose.Res always stands at the end of an SML response file. It terminates this file.

```
(J)   SML_PublicClose.Res      ::=    SEQUENCE
      {
        globalSignature                SML_Signature OPTIONAL
      }

(K)   SML_Signature            ::= Octet String
```

6.1.5 SML_Attention.Res

SML_Attention.Res is used in an SML response file to report potentially positive acknowledgements, error messages, warnings or other notes from the server to the client.

To enable the (error) messages to be both automatically evaluated and simply outputted as text to the operator, the ‘attentionNo’ and ‘attentionMsg’ elements are used.

Assignment of globally valid numbers is defined in Section 6.1.5.1.

```
(L)   SML_Attention.Res      ::=   SEQUENCE
      {
        serverId                   Octet String,
        attentionNo                Octet String,
        attentionMsg               Octet String OPTIONAL,
        attentionDetails           SML_Tree OPTIONAL
      }
```

6.1.5.1 Global SML Attention Numbers

The list of globally defined Error Numbers is provided below, see Section 6.1.5:

Tab. 2: List of global Error Numbers

Error Number (displayed as a byte chain in hexadecimal form)	Significance
…	… reserved.
81 81 C7 C7 E0 00	Beginning of application-specific Error Numbers
…	Application-specific Error Numbers
81 81 C7 C7 FC FF	End of application-specific Error Numbers
81 81 C7 C7 FD 00	See T
…	See T
81 81 C7 C7 FD FF	See T
81 81 C7 C7 FE 00	Error message that cannot be assigned to any of the meanings defined below.
81 81 C7 C7 FE 01	Unknown SML designator.
81 81 C7 C7 FE 02	Inadequate authentication, user/password combination impermissible.
81 81 C7 C7 FE 03	Destination address (‘serverId’)not available.
81 81 C7 C7 FE 04	Order (‘reqFileId’) not available.
81 81 C7 C7 FE 05	One or more destination attribute(s) cannot be written.
81 81 C7 C7 FE 06	One or more destination attribute(s) cannot be read.
81 81 C7 C7 FE 07	Communication with measuring point disturbed.
81 81 C7 C7 FE 08	Raw data cannot be interpreted.
81 81 C7 C7 FE 09	Value supplied is outside the permissible value range.
81 81 C7 C7 FE 0A	Order not executed (for example, because the ‘parameterTreePath’ supplies indicates an absent element.
81 81 C7 C7 FE 0B	Checksum faulty
81 81 C7 C7 FE 0C	Broadcast not supported
81 81 C7 C7 FE 0D	Unexpected SML message (e.g. an SML file without an Open Request)
81 81 C7 C7 FE 0E	Unknown object in the profile (the OBIS code in the request for a profile refers to a data source not recorded in the profile)
81 81 C7 C7 FE 0F	Data type not supported (e.g. the data type in a SetProcPar.Req message does not correspond to the expected data type)
81 81 C7 C7 FE 10	Optional element not supported (an element defined in SML as OPTIONAL, which is not supported by the application, has been received as OPTIONAL.)
81 81 C7 C7 FE 11	No entry in requested profile
81 81 C7 C7 FE 12	In profile requests: end limit before begin limit
81 81 C7 C7 FE 13	In profile requests: No entries in requested area. At least one entry in other areas.
81 81 C7 C7 FE 14	An SML file was terminated without SML-Close.

Error Number (displayed as a byte chain in hexadecimal form)	Significance
81 81 C7 C7 FE 15	In profile requests: Temporarily the profile cannot be sent (e.g. due to reordering at request moment or a signature for the profile entry has to be calculated). In general: The response cannot be sent ("busy").
…	… reserved.

The list of globally defined Attention Numbers is given below, see Section 6.1.5:

Tab. 3: List of global Attention Numbers

Attention Number (displayed as a byte chain in hexadecimal form)	Significance
81 81 C7 C7 FD 00	OK, positive acknowledgement.
81 81 C7 C7 FD 01	Order will be executed later and result passed to server address per Response-without-Request.
…	… reserved

6.1.6 SML_GetCosem.Req

05.05.2011: Instead of 3 COSEM services one generic COSEM service with reference on the green book could be a solution?

Per SML_GetCosem.Req the COSEM service ‚Get' is transmitted. As an answer either an SML_GetCosem.Res oder ein SML_Attention.Res must be generated. The empty 'ServerID' must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(M)   SML_GetCosem.Req      ::=   SEQUENCE
      {
        clientId                  Octet String,
        serverId                  Octet String OPTIONAL,
        username                  Octet String OPTIONAL,
        password                  Octet String OPTIONAL,

      → 05.05.11: Use a reference to the corresponding COSEM service

        logicalName               Octet String,
        classId                   Integer16,
        classVersion              Integer16,
        attributeIndexList        SML_CosemAttrIndexList OPTIONAL
      }
```

```
(N)  SML_CosemAttrIndexList   ::=   SEQUENCE OF
     {
       attributeDescription       SML_CosemAttributeDesc
     }

(O)  SML_CosemAttributeDesc   ::=   SEQUENCE
     {
       attributeIndex             Integer16,
       selectiveAccessDescriptor  SML_CosemSelAccessDesc OPTIONAL
     }

(P)  SML_CosemSelAccessDesc::=   SEQUENCE OF
     {
       accessSelector             Unsigned8,
       accessParameters           SML_CosemValue
     }
```

If the element ‘attributeIndexList’ is not given then all attributes of the addressed object must be delivered.

6.1.7 SML_GetCosem.Res

Per SML_GetCosem.Res the answer to a COSEM service ‚Get’ is transmitted.

```
(Q)  SML_GetCosem.Res         ::=   SEQUENCE
     {
       clientId                   Octet StringOPTIONAL,
       serverId                   Octet String,
       logicalName                Octet String,
       classId                    Integer16,
       classVersion               Integer16,
       attributeList              SML_CosemAttrList
     }

(R)  SML_CosemAttrList        ::=   SEQUENCE OF
     {
       cosemAttribute             SML_CosemAttribute
     }

(S)  SML_CosemAttribute       ::=   SEQUENCE
     {
       attributeDescription       SML_CosemAttributeDesc
       attributeContent           SML_CosemAttributeContent
     }

(T)  SML_CosemAttributeContent      ::= CHOICE
     {
       data                       [0x01]    SML_CosemValue
       dataAccessResult           [0x02]    Unsigned8,
     }
```

```
(U)  SML_CosemValueList        ::= SEQUENCE OF
     {
       cosemValue                      SML_CosemValue
     }

(V)  SML_CosemValue            ::= CHOICE
     {
       implicit-data-type              [0x01]  SML_CosemValueSimple,
       nullData                        [0x02]  Octet String (SIZE(0)),
       dont-care                       [0x03]  Octet String (SIZE(0)),
       bit-string                      [0x04]  Octet String
                                               (leading '0'-bits are used, if the bit-string
                                               length does not fit inside a complete byte),
       floating-point                  [0x05]  Octet String (SIZE(4)),
       visible-string                  [0x06]  Octet String,
       bcd                             [0x07]  Integer8,
       enum                            [0x08]  Integer8,
       float32                         [0x09]  Octet String (SIZE(4)),
       float64                         [0x0A]  Octet String (SIZE(8)),
       date_time                       [0x0b]  SML_Time,
       date                            [0x0C]  SML_Time,
       time                            [0x0d]  SML_Time,
       array                           [0x0E]  SML_CosemValueList,
       compact-array                   [0x0F]  SML_CosemCompactArray
     }

(W)  SML_CosemValueSimple      ::= IMPLICIT CHOICE
     {
       boolean-Value                   boolean,
       byte-List                       Octet String,
       integer                         Integer8,
       long                            Integer16,
       double-long                     Integer32,
       long64                          Integer64,
       unsigned                        Unsigned8,
       long-unsigned                   Unsigned16,
       double-long-unsigned            Unsigned32,
       long64-unsigned                 Unsigned64,
       structure                       SML_CosemValueList
     }

(X)  SML_CosemCompactArray     ::= SEQUENCE OF
     {
       contents-description            SML_CosemTypeDef,
       array-contents                  Octet String
     }
```

```
(Y)   SML_CosemTypeDef          ::=   CHOICE
      {
        null-data                     [0x01]   Octet String (SIZE(0)),
        array                         [0x02]   SML_CosemArrayDef,
        structure                     [0x03]   SML_CosemTypeList,
        boolean                       [0x04]   Octet String (SIZE(0)),
        bit-string                    [0x05]   Octet String (SIZE(0)),
        double-long                   [0x06]   Octet String (SIZE(0)),
        double-long-unsigned          [0x07]   Octet String (SIZE(0)),
        floating-point                [0x08]   Octet String (SIZE(0)),
        octet-string                  [0x09]   Octet String (SIZE(0)),
        visible-string                [0x0A]   Octet String (SIZE(0)),
        bcd                           [0x0b]   Octet String (SIZE(0)),
        integer                       [0x0C]   Octet String (SIZE(0)),
        long                          [0x0d]   Octet String (SIZE(0)),
        unsigned                      [0x0E]   Octet String (SIZE(0)),
        long-unsigned                 [0x0F]   Octet String (SIZE(0)),
        long64                        [0x10]   Octet String (SIZE(0)),
        long64-unsigned               [0x11]   Octet String (SIZE(0)),
        enum                          [0x12]   Octet String (SIZE(0)),
        float32                       [0x13]   Octet String (SIZE(0)),
        float64                       [0x14]   Octet String (SIZE(0)),
        date_time                     [0x15]   Octet String (SIZE(0)),
        date                          [0x16]   Octet String (SIZE(0)),
        time                          [0x17]   Octet String (SIZE(0)),
        dont-care                     [0x18]   Octet String (SIZE(0)),
      }

(Z)   SML_CosemArrayDef         ::=   SEQUENCE
      {
        array-element                 SML_CosemArrayElement
      }

(AA)  SML_CosemArrayElement     ::=   SEQUENCE OF
      {
        number-of-elements            Unsigned16,
        type-description              SML_CosemTypeDef
      }

(BB)  SML_CosemTypeList         ::=   SEQUENCE
      {
        type-description              SML_CosemTypeDef
      }
```

6.1.8 SML_SetCosem.Req

Per SML_SetCosem.Req the COSEM service ‚Set' is transmitted. As an answer either a SML_SetCosem.Res or a SML_Attention.Res must be generated.The empty 'ServerID' must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(CC)  SML_SetCosem.Req        ::=   SEQUENCE
      {
        clientId                    Octet String,
        serverId                    Octet StringOPTIONAL,
        username                    Octet StringOPTIONAL,
        password                    Octet StringOPTIONAL,
        objName                     Octet String,
        classId                     Integer16,
        classVersion                Integer16,
        attributeList               SML_CosemAttrList
      }
```

6.1.9 SML_SetCosem.Res

Per SML_SetCosem.Res the answer to a COSEM service ‚Set' is transmitted.

```
(DD)  SML_SetCosem.Res        ::=   SEQUENCE
      {
        clientId                    Octet StringOPTIONAL,
        serverId                    Octet String,
        objName                     Octet String,
        classId                     Integer16,
        classVersion                Integer16,
        attributeList               SML_CosemAttrList OPTIONAL
      }
```

6.1.10 SML_ActionCosem.Req

Per SML_ActionCosem.Req the COSEM service ‚Execute' is transmitted. As an answer either a SML_ActionCosem.Res or a SML_Attention.Res must be generated. The empty 'ServerID' must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(EE)  SML_ActionCosem.Req     ::=   SEQUENCE
      {
        clientId                    Octet String,
        serverId                    Octet StringOPTIONAL,
        username                    Octet StringOPTIONAL,
        password                    Octet StringOPTIONAL,
        objName                     Octet String,
        classId                     Integer16,
        classVersion                Integer16,
        serviceIndex                Unsigned8,
        serviceParameter            SML_CosemValue  OPTIONAL
      }
```

6.1.11 SML_ActionCosem.Res

Per SML_ActionCosem.Res the answer to a COSEM service ‚Execute' is transmitted.

```
(FF)  SML_ActionCosem.Res  ::=  SEQUENCE
      {
        clientId                Octet String OPTIONAL,
        serverId                Octet String,
        objName                 Octet String,
        classId                 Integer16,
        classVersion            Integer16,
        attributeList           SML_CosemAttrList  OPTIONAL
      }
```

7 SML binary encoding, directly packed coding

SML uses a coding optimised for the goal of producing minimised messages in the data flow.

The coding for this purpose is based on the classical type-length-value structure. In contrast to BER, however, it groups together type and length in a value known as the type-length field ("TL-Field") reduced to a single byte for most application cases.

The length field quantifies the number of elements that belong to a simple or complex data type.

In the case of simple data types, the length particular corresponds to the number of bytes that belong to the data type concerned.

Since the length field is a part of the code, and the TL-Field itself can be regarded as an element, the TL-Field is also counted as a further element in the length particular.

In the case of complex data types (such as lists), the length particular corresponds to the number of elements that are grouped together with the complex data type (for example, the number of list entries). The TL-Field itself is not counted here.

7.1 Type-length field

The TL-Field uses the bit combination in the higher-valued bits of the data word to specify whether, and if so with what significance, further bytes shall be put together with the current byte to form a single word.

Tab. 4: Bit coding in the type-length field for the first byte of a TL-Field particular

Bit index	MSB, D7	6	5	4	3	2	1	LSB, D0
Used for feature "Further byte for TL-follows'	1	X	X	X	X	X	X	X
Used for feature 'No further byte for TL-Field follows'	0	X	X	X	X	X	X	X
Used for feature 'Data type Octet String'	X	0	0	0	L	L	L	L
Used for feature 'Boolean'	0	1	0	0	L	L	L	L
Used for feature "Data type integer'	X	1	0	1	L	L	L	L
Used for feature "'Data type	X	1	1	0	L	L	L	L

Bit index	MSB, D7	6	5	4	3	2	1	LSB, D0
unsigned'								
Used for feature 'Data type List of …'.	X	1	1	1	L	L	L	L
Further byte with space for defining additional data types, currently reserved	1	1	0	0	L	L	L	L
Reserved for future use	X	0	0	1	L	L	L	L
Reserved for future use	X	0	1	0	L	L	L	L
Reserved for future use	X	0	1	1	L	L	L	L

If a second (and perhaps further) byte(s) follow(s) with a TL-Field particular, the bits for the length information of the TL-Field particular preceding in each case will be shifted to the left and the "new" bits of the following TL-Field particular added coming from the right.

For the second and any following additional bytes with a TL-Field particular, the bits for the data type must always be set as follows:

Tab. 5: Bit coding in the type-length field for the second and following TL-Field bytes

Bit index	MSB, D7	6	5	4	3	2	1	LSB, D0
Used for feature 'Further byte for TL-Field follows'	1	X	X	X	X	X	X	X
Used for feature "No further byte for TL-Field follows'	0	X	X	X	X	X	X	X
Feature 'Use following 4 bits for the length'	X	0	0	0	L	L	L	L
Reserved for future purposes	X	0	0	1	X	X	X	X
Reserved for future purposes	X	0	1	0	X	X	X	X
Reserved for future purposes	X	0	1	1	X	X	X	X
Reserved for future purposes	X	1	X	X	X	X	X	X

7.2 Coding of data types

7.2.1 Data type Octet String

An Octet String (a byte chain) with a length of 0 to max. 14 bytes is coded as follows:

Tab. 6: Bit coding in type-length field for an Octet String

Bit index	MSB, D7	6	5	4	3	2	1	LSB, D0
Octet String with a number from 0 to 14 bytes	0	0	0	0	L	L	L	L

The TL-Field is followed by the Octet String (the byte chain), where the byte of the byte chain with the index '0' has to follow first behind the TL-Field.

If the Octet String contains more than 14 bytes, then in accordance with the description in Section 7.1 further bytes of the TL-Field are inserted before the first byte of the byte chain.

7.2.2 Data types Integer8, Integer16, Integer32 and Integer64

These integer data types are coded as follows; note that in each data transmission entire bytes that contain leading zeros (in the case of positive numbers) or leading ones (in the case of negative numbers), may be omitted in such a way that no falsification is produced at the recipient:

```
(GG)  Integer8                ::= SEQUENCE
      {
        TL-Field              0x52,
        Data value            0xYY
      }

(HH)  Integer16               ::= SEQUENCE
      {
        TL-Field              0x53,
        Data value            0xYY 0xZZ (0xYY ⇔ High-Byte,
                                         0xZZ ⇔ Low-Byte,
                                         Big Endian representation)
      }

(II)  Integer32               ::= SEQUENCE
      {
        TL-Field              0x55,
        Data value            0xYY 0xZZ 0xUU 0xVV   (0xYY ⇔ High-Byte,
                                                    0xVV ⇔ Low-Byte,
                                                    Big Endian representation)
      }

(JJ)  Integer64               ::= SEQUENCE
      {
        TL-Field              0x59,
        Data value            0xYY ... 0xVV      (0xYY ⇔ High-Byte,
                                                  0xVV ⇔ Low-Byte,
                                                  Big Endian representation)
      }
```

7.2.3 Data types Unsigned8, Unsigned16, Unsigned32 and Unsigned64

These unsigned integer data types are coded as follows; note that in each data transmission entire bytes that contains leading zeroes may be omitted:

```
(KK)  Unsigned8               ::= SEQUENCE
      {
        TL-Field              0x62,
        Data value            0xYY
      }
```

```
(LL)  Unsigned16          ::=  SEQUENCE
      {
        TL-Field                0x63,
        Data value              0xYY 0xZZ (0xYY ⇔ High-Byte,
                                           0xZZ ⇔ Low-Byte,
                                           depiction Big Endian)
      }

(MM)  Unsigned32          ::=  SEQUENCE
      {
        TL-Field                0x65,
        Data value              0xYY 0xZZ 0xUU 0xVV   (0xYY ⇔ High-Byte,
                                                      0xVV ⇔ Low-Byte,
                                                      depiction Big Endian)
      }

(NN)  Unsigned64          ::=  SEQUENCE
      {
        TL-Field                0x69,
        Data value              0xYY ... 0xVV        (0xYY ⇔ High-Byte,
                                                      0xVV ⇔ Low-Byte,
                                                      depiction Big Endian)
      }
```

7.2.4 Data type Boolean

This Boolean data type is coded as follows:

```
(OO)  Boolean             ::=  SEQUENCE
      {
        TL-Field                0x42,
        Data value              0xYY        (        0x00 ⇔ 'false',
      }                                       everything else ⇔ 'true' )
```

7.2.5 Data type List of ...

If lists (or arrays or structures) are to be coded, the TL field shall contain at the beginning of the list/array/structure the number of elements. This is followed by the first element of the list/array/structure, which likewise again begins with a TL-Field.

The element particular in the TL-Field of 'List of ...' always refers to the next TL-Field, which follows behind all elements of 'List of ...'. The value thus 'leaps' over all TL-Fields that for each element of 'List of ...' stand inside the 'List of ...' data.

```
(PP)  List of ...         ::=  SEQUENCE
      {
        TL-Field                0x7z        ('z' ⇔ Number of elements
                                             in the list)
      }
```

7.3 Coding of special features

7.3.1 Feature End of an SML message

Due to concatenation of the individual attributes of an SML message, the end of an SML message can be defined/found using the TL-Field:

(QQ) EndOfSmlMsg ::= 0x00 ⇔ “Entry without TL-Field“ ⇔ End

7.3.2 Feature SEQUENCE

In the case of sequences, all components of the sequence are always accepted into the coded data flow in the order in which they are listed in the ASN.1 definition.

A sequence is always grouped together as a structure, and thus initiated by the data type ‘List of ...’ (see Section 7.2.5).

7.3.3 Feature CHOICE

In the case of selection lists, the TAG is coded to the selected element Unsigned. The TAG contains, just like any other unsigned component, its TL-Field as a prefix.

Elements of the type SML_MessageBody code the TAG as Unsigned32, all others as Unsigned8.

The CHOICE itself is treated as a structure and is thus initiated by the data type ‘List of ...’ (see Section 7.2.5). Thus it always leads to a structure with two elements: the first element is the TAG and the second the thereby defined CHOICE element.

7.3.4 Feature OPTIONAL

If in the ASN.1 definition a component has been designated as ‘OPTIONAL’, it must be set in the data flow with the TL-Field ‘0x01’. Note that for this TL-Field the data type ‘Octet String’ is always assumed, and the element particular positioned directly behind the TL-Field.

8 XML - coding

Alternatively SML can also be transported per XML coding. Compared with the method described in Section 7, XML coding basically entails a significantly increased data volume, but supplies SML files in standardised readable form.

In the following the XML scheme for coding of SML per XML is concerted.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault     = "qualified"
           xmlns:xs               = "http://www.w3.org/2001/XMLSchema">


<!-- Simple data types                                                                    -->
<!-- ================================================================================== -->

<xs:simpleType          name = "SML_Timestamp">
    <xs:restriction     base = "xs:dateTime"/>
```

```
</xs:simpleType>

<xs:simpleType          name = "SML_Signature">
    <xs:restriction     base = "xs:hexBinary"/>
</xs:simpleType>

<xs:simpleType          name = "SML_ObjReqEntry">
    <xs:restriction     base = "xs:hexBinary"/>
</xs:simpleType>

<xs:simpleType          name = "SML_Unit">
    <xs:restriction     base = "xs:unsignedByte"/>
</xs:simpleType>



<!-- Implicite selection data types                                                                    -->
<!-- ================================================================================================ -->

<xs:group               name = "SML_Value">
    <xs:choice>
        <xs:element     name = "valBoolean"      type = "xs:boolean"/>
        <xs:element     name = "valByteList"     type = "xs:hexBinary"/>
        <xs:element     name = "valInteger8"     type = "xs:byte"/>
        <xs:element     name = "valInteger16"    type = "xs:short"/>
        <xs:element     name = "valInteger32"    type = "xs:int"/>
        <xs:element     name = "valInteger64"    type = "xs:long"/>
        <xs:element     name = "valUnsigned8"    type = "xs:unsignedByte"/>
        <xs:element     name = "valUnsigned16"   type = "xs:unsignedShort"/>
        <xs:element     name = "valUnsigned32"   type = "xs:unsignedInt"/>
        <xs:element     name = "valUnsigned64"   type = "xs:unsignedLong"/>
    </xs:choice>
</xs:group>

<xs:group               name = "SML_Status">
    <xs:choice>
        <xs:element     name = "status8"         type = "xs:unsignedByte"/>
        <xs:element     name = "status16"        type = "xs:unsignedShort"/>
        <xs:element     name = "status32"        type = "xs:unsignedInt"/>
        <xs:element     name = "status64"        type = "xs:unsignedLong"/>
    </xs:choice>
</xs:group>



<!--Complex data types                                                                            -->
<!-- ================================================================================================ -->

<xs:complexType         name = "SML_TimestampLocal">
    <xs:sequence>
        <xs:element     name = "timestamp"            type = "SML_Timestamp"/>
        <xs:element     name = "localOffset"          type = "xs:short"/>
        <xs:element     name = "summerTimeOffset"     type = "xs:short"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_Time">
    <xs:choice>
        <xs:element     name = "secIndex"             type = "xs:unsignedInt"/>
        <xs:element     name = "timestamp"            type = "SML_Timestamp"/>
        <xs:element     name = "localTimestamp"       type = "SML_TimestampLocal"/>
    </xs:choice>
</xs:complexType>

<xs:complexType         name = "SML_ProfObjHeaderEntry">
    <xs:sequence>
        <xs:element     name = "objName"    type = "xs:hexBinary"/>
        <xs:element     name = "unit"       type = "SML_Unit"/>
        <xs:element     name = "scaler"     type = "xs:byte"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_ProfObjPeriodEntry">
    <xs:sequence>
        <xs:element     name = "valTime"         type      = "SML_Time"/>
        <xs:element     name = "status"          type      = "xs:unsignedLong"/>
        <xs:element     name = "value_List"      type      = "List_of_SML_ValueEntry"/>
        <xs:element     name = "periodSignature" type      = "SML_Signature"
                                                 minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_ValueEntry">
    <xs:sequence>
        <xs:group       ref  = "SML_Value" />
        <xs:element     name = "valueSignature"  type      = "SML_Signature"
                                                 minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType         name = "SML_PeriodEntry">
    <xs:sequence>
        <xs:element     name = "objName"         type      = "xs:hexBinary"/>
        <xs:element     name = "unit"            type      = "SML_Unit"/>
        <xs:element     name = "scaler"          type      = "xs:byte"/>
        <xs:group       ref  = "SML_Value" />
        <xs:element     name = "valueSignature"  type      = "SML_Signature"
                                                 minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType             name = "SML_ProcParValue">
    <xs:choice>
        <xs:element         name = "smlValue">
            <xs:complexType>
                <xs:group   ref  = "SML_Value"/>
            </xs:complexType>
        </xs:element>
        <xs:element         name = "smlPeriodEntry"  type = "SML_PeriodEntry"/>
        <xs:element         name = "smlTupelEntry"   type = "SML_TupelEntry"/>
        <xs:element         name = "smlTime"         type = "SML_Time"/>
        <xs:element         name = "smlListEntry"    type = "SML_ListEntry"/>
    </xs:choice>
</xs:complexType>

<xs:complexType         name = "SML_TupelEntry">
    <xs:sequence>
        <xs:element     name = "serverId"             type = "xs:hexBinary"/>
        <xs:element     name = "secIndex"             type = "SML_Time"/>
        <xs:element     name = "status"               type = "xs:unsignedLong"/>
        <xs:element     name = "unit_pA"              type = "SML_Unit"/>
        <xs:element     name = "scaler_pA"            type = "xs:byte"/>
        <xs:element     name = "value_pA"             type = "xs:long"/>
        <xs:element     name = "unit_R1"              type = "SML_Unit"/>
        <xs:element     name = "scaler_R1"            type = "xs:byte"/>
        <xs:element     name = "value_R1"             type = "xs:long"/>
        <xs:element     name = "unit_R4"              type = "SML_Unit"/>
        <xs:element     name = "scaler_R4"            type = "xs:byte"/>
        <xs:element     name = "value_R4"             type = "xs:long"/>
        <xs:element     name = "signature_pA_R1_R4"   type = "xs:hexBinary"/>
        <xs:element     name = "unit_mA"              type = "SML_Unit"/>
        <xs:element     name = "scaler_mA"            type = "xs:byte"/>
        <xs:element     name = "value_mA"             type = "xs:long"/>
        <xs:element     name = "unit_R2"              type = "SML_Unit"/>
        <xs:element     name = "scaler_R2"            type = "xs:byte"/>
        <xs:element     name = "value_R2"             type = "xs:long"/>
        <xs:element     name = "unit_R3"              type = "SML_Unit"/>
        <xs:element     name = "scaler_R3"            type = "xs:byte"/>
        <xs:element     name = "value_R3"             type = "xs:long"/>
        <xs:element     name = "signature_mA_R2_R3"   type = "xs:hexBinary"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_TreePath">
    <xs:sequence>
        <xs:element     name = "path_Entry"      type      = "xs:hexBinary"
                                                 minOccurs = "1"
                                                 maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType             name = "SML_Tree">
    <xs:sequence>
        <xs:element         name = "parameterName"   type      ="xs:hexBinary"/>
        <xs:element         name = "parameterValue"  type      = "SML_ProcParValue"
                                                     minOccurs = "0"
                                                     maxOccurs = "1"/>
        <xs:element         name = "child_List"      type      = "List_of_SML_Tree"
                                                     minOccurs = "0"
                                                     maxOccurs = "1"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType             name = "SML_ListEntry">
    <xs:sequence>
        <xs:element         name = "objName"         type      = "xs:hexBinary"/>
        <xs:group           ref  = "SML_Status"      minOccurs = "0"/>
        <xs:element         name = "valTime"         type      = "SML_Time"
                                                     minOccurs = "0"/>
        <xs:element         name = "unit"            type      = "SML_Unit"
                                                     minOccurs = "0"/>
        <xs:element         name = "scaler"          type      = "xs:byte"
                                                     minOccurs = "0"/>
        <xs:group           ref  = "SML_Value"/>
        <xs:element         name = "valueSignature"  type      = "SML_Signature"
                                                     minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>
```

```
<!--List data types                                                                          -->
<!-- ========================================================================================= -->

<xs:complexType         name = "List_of_SML_ValueEntry">
    <xs:sequence>
        <xs:element     name = "value_List_Entry"     type      = "SML_ValueEntry"
                                                      minOccurs = "0"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "List_of_SML_ProfObjHeaderEntry">
    <xs:sequence>
        <xs:element     name = "header_List_Entry"    type      = "SML_ProfObjHeaderEntry"
                                                      minOccurs = "0"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "List_of_SML_ProfObjPeriodEntry">
    <xs:sequence>
        <xs:element     name = "period_List_Entry"    type      = "SML_ProfObjPeriodEntry"
                                                      minOccurs = "0"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "List_of_SML_PeriodEntry">
    <xs:sequence>
        <xs:element     name = "period_List_Entry"    type      = "SML_PeriodEntry"
                                                      minOccurs = "0"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "List_of_SML_Tree">
    <xs:sequence>
        <xs:element     name = "tree_Entry"           type      = "SML_Tree"
                                                      minOccurs = "1"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_List">
    <xs:sequence>
        <xs:element     name = "valListEntry"         type      = "SML_ListEntry"
                                                      minOccurs = "0"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "List_of_SML_ObjReqEntry">
    <xs:sequence>
        <xs:element     name = "object_List_Entry"    type      = "SML_ObjReqEntry"
                                                      minOccurs = "1"
                                                      maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>



<! --SML messages                                                                          -->
<!-- ========================================================================================= -->

     <!-- SML-Open                                                                          -->
     <!-- ..................................................................................... -->

<xs:complexType         name = "SML_PublicOpen.Req">
    <xs:sequence>
        <xs:element     name = "codepage"         type = "xs:hexBinary"         minOccurs = "0"/>
        <xs:element     name = "clientId"         type = "xs:hexBinary"/>
        <xs:element     name = "reqFileId"        type = "xs:hexBinary"/>
        <xs:element     name = "serverId"         type = "xs:hexBinary"         minOccurs = "0"/>
        <xs:element     name = "username"         type = "xs:hexBinary"         minOccurs = "0"/>
        <xs:element     name = "password"         type = "xs:hexBinary"         minOccurs = "0"/>
        <xs:element     name = "smlVersion"       type = "xs:unsignedByte"      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_PublicOpen.Res">
    <xs:sequence>
        <xs:element     name = "codepage"         type = "xs:hexBinary"         minOccurs = "0"/>
        <xs:element     name = "clientId"         type = "xs:hexBinary"         minOccurs = "0"/>
        <xs:element     name = "reqFileId"        type = "xs:hexBinary"/>
        <xs:element     name = "serverId"         type = "xs:hexBinary"/>
        <xs:element     name = "refTime"          type = "SML_Time"             minOccurs = "0"/>
        <xs:element     name = "smlVersion"       type = "xs:unsignedByte"      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>
```

```
     <!-- SML-Close                                                                  -->
     <!-- ------------------------------------------------------------------------- -->

<xs:complexType         name = "SML_PublicClose.Req">
    <xs:sequence>
        <xs:element     name = "globalSignature" type = "SML_Signature"       minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_PublicClose.Res">
    <xs:sequence>
        <xs:element     name = "globalSignature" type = "SML_Signature"       minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>



     <!-- SML-GetProfilePack                                                         -->
     <!-- ------------------------------------------------------------------------- -->

<xs:complexType         name = "SML_GetProfilePack.Req">
    <xs:sequence>
        <xs:element     name = "serverId"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "username"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "password"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "withRawdata"         type      = "xs:boolean"
                                                     minOccurs = "0"/>
        <xs:element     name = "beginTime"           type      = "SML_Time"
                                                     minOccurs = "0"/>
        <xs:element     name = "endTime"             type      = "SML_Time"
                                                     minOccurs = "0"/>
        <xs:element     name = "parameterTreePath"   type      = "SML_TreePath"/>
        <xs:element     name = "object_List"         type      = "List_of_SML_ObjReqEntry"
                                                     minOccurs = "0"/>
        <xs:element     name = "dasDetails"          type      = "SML_Tree"
                                                     minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_GetProfilePack.Res">
    <xs:sequence>
        <xs:element     name = "serverId"            type      = "xs:hexBinary"/>
        <xs:element     name = "actTime"             type      = "SML_Time"/>
        <xs:element     name = "regPeriod"           type      = "xs:unsignedInt"/>
        <xs:element     name = "parameterTreePath"   type      = "SML_TreePath"/>
        <xs:element     name = "header_List"         type      = "List_of_SML_ProfObjHeaderEntry"/>
        <xs:element     name = "period_List"         type      = "List_of_SML_ProfObjPeriodEntry"/>
        <xs:element     name = "rawdata"             type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "profileSignature"    type      = "SML_Signature"
                                                     minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>



     <!-- SML-GetProfileList                                                         -->
     <!-- ------------------------------------------------------------------------- -->

<xs:complexType         name = "SML_GetProfileList.Req">
    <xs:sequence>
        <xs:element     name = "serverId"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "username"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "password"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "withRawdata"         type      = "xs:boolean"
                                                     minOccurs = "0"/>
        <xs:element     name = "beginTime"           type      = "SML_Time"
                                                     minOccurs = "0"/>
        <xs:element     name = "endTime"             type      = "SML_Time"
                                                     minOccurs = "0"/>
        <xs:element     name = "parameterTreePath"   type      = "SML_TreePath"/>
        <xs:element     name = "object_List"         type      = "List_of_SML_ObjReqEntry"
                                                     minOccurs = "0"/>
        <xs:element     name = "dasDetails"          type      = "SML_Tree"
                                                     minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType         name = "SML_GetProfileList.Res">
    <xs:sequence>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"/>
        <xs:element     name = "actTime"              type      = "SML_Time"/>
        <xs:element     name = "regPeriod"            type      = "xs:unsignedInt"/>
        <xs:element     name = "parameterTreePath"    type      = "SML_TreePath"/>
        <xs:element     name = "valTime"              type      = "SML_Time"/>
        <xs:element     name = "status"               type      = "xs:unsignedLong"/>
        <xs:element     name = "period_List"          type      = "List_of_SML_PeriodEntry"/>
        <xs:element     name = "rawdata"              type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "periodSignature"      type      = "SML_Signature"
                                                      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>



     <!-- SML-GetProcParameter                                                          -->
     <!-- ..................................................................... -->

<xs:complexType         name = "SML_GetProcParameter.Req">
    <xs:sequence>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "username"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "password"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "parameterTreePath"    type      = "SML_TreePath"/>
        <xs:element     name = "attribute"            type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_GetProcParameter.Res">
    <xs:sequence>
        <xs:element     name = "serverId"             type = "xs:hexBinary"/>
        <xs:element     name = "parameterTreePath"    type = "SML_TreePath"/>
        <xs:element     name = "parameterTree"        type = "SML_Tree"/>
    </xs:sequence>
</xs:complexType>



     <!-- SML-SetProcParameter                                                          -->
     <!--     (Here only one request is possible)                                          -->
     <!-- ..................................................................... -->

<xs:complexType         name = "SML_SetProcParameter.Req">
    <xs:sequence>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "username"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "password"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "parameterTreePath"    type      = "SML_TreePath"/>
        <xs:element     name = "parameterTree"        type      = "SML_Tree"/>
    </xs:sequence>
</xs:complexType>



     <!-- SML-GetList                                                                   -->
     <!-- ..................................................................... -->

<xs:complexType         name = "SML_GetList.Req">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"
                                                      minOccurs = "0" />
        <xs:element     name = "username"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "password"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "listName"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_GetList.Res">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"/>
        <xs:element     name = "listName"             type      = "xs:hexBinary"
```

```
                                                    minOccurs = "0"/>
        <xs:element     name = "actSensorTime"      type      = "SML_Time"
                                                    minOccurs = "0"/>
        <xs:element     name = "valList"            type      = "SML_List"/>
        <xs:element     name = "listSignature"      type      = "SML_Signature"
                                                    minOccurs = "0"/>
        <xs:element     name = "actGatewayTime"     type      = "SML_Time"
                                                    minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>




     <!-- SML-Attention                                                              -->
     <!--     (Here only one response is possible)                                      -->
     <!-- ------------------------------------------------------------------------- -->

<xs:complexType         name = "SML_Attention.Res">
    <xs:sequence>
        <xs:element     name = "serverId"           type      = "xs:hexBinary"/>
        <xs:element     name = "attentionNo"        type      = "xs:hexBinary"/>
        <xs:element     name = "attentionMsg"       type      = "xs:string"
                                                    minOccurs = "0"/>
        <xs:element     name = "attentionDetails"   type      = "SML_Tree"
                                                    minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>




     <!-- SML-Message-Body                                                           -->
     <!-- ------------------------------------------------------------------------- -->

<xs:complexType         name = "SML_Message">
    <xs:sequence>
        <xs:element     name = "transactionId"          type = "xs:hexBinary"/>
        <xs:element     name = "groupNo"                type = "xs:unsignedByte"/>
        <xs:element     name = "abortOnError"           type = "xs:unsignedByte"/>
        <xs:choice>
            <xs:element name = "OpenRequest"            type = "SML_PublicOpen.Req"/>
            <xs:element name = "OpenResponse"           type = "SML_PublicOpen.Res"/>
            <xs:element name = "CloseRequest"           type = "SML_PublicClose.Req"/>
            <xs:element name = "CloseResponse"          type = "SML_PublicClose.Res"/>
            <xs:element name = "GetProfilePackRequest"  type = "SML_GetProfilePack.Req"/>
            <xs:element name = "GetProfilePackResponse" type = "SML_GetProfilePack.Res"/>
            <xs:element name = "GetProfileListRequest"  type = "SML_GetProfileList.Req"/>
            <xs:element name = "GetProfileListResponse" type = "SML_GetProfileList.Res"/>
            <xs:element name = "GetProcParameterRequest"  type = "SML_GetProcParameter.Req"/>
            <xs:element name = "GetProcParameterResponse" type = "SML_GetProcParameter.Res"/>
            <xs:element name = "SetProcParameterRequest"  type = "SML_SetProcParameter.Req"/>
            <xs:element name = "GetListRequest"         type = "SML_GetList.Req"/>
            <xs:element name = "GetListResponse"        type = "SML_GetList.Res"/>
            <xs:element name = "GetCosemRequest"        type = "SML_GetCosem.Req"/>
            <xs:element name = "GetCosemResponse"       type = "SML_GetCosem.Res"/>
            <xs:element name = "SetCosemRequest"        type = "SML_SetCosem.Req"/>
            <xs:element name = "SetCosemResponse"       type = "SML_SetCosem.Res"/>
            <xs:element name = "RunCosemRequest"        type = "SML_RunCosem.Req"/>
            <xs:element name = "RunCosemResponse"       type = "SML_RunCosem.Res"/>
            <xs:element name = "AttentionResponse"      type = "SML_Attention.Res"/>
        </xs:choice>
    </xs:sequence>
</xs:complexType>



<!-- SML file                                                                     -->
<!-- ========================================================================== -->

<xs:complexType         name      = "SML_File">
    <xs:sequence>
        <xs:element     name      = "SmlMessage" type      = "SML_Message"
                        minOccurs = "1"          maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>



<!-- XML root elements                                                          -->
<!-- ========================================================================== -->

<xs:complexType         name      = "SML_XML_Body_V_1_04">
    <xs:choice>
        <xs:element     name      = "smlFile"    type = "SML_File"/>
        <xs:element     name      = "smlMessage" type = "SML_Message"/>
    </xs:choice>
</xs:complexType>
```

```
<xs:element             name      = "SML_XML_1_04">
    <xs:complexType>
        <xs:sequence>
            <xs:element name      = "smlXmlBody" type      = "SML_XML_Body_V_1_04"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>



<!--COSEM specific data types                                                                  -->
<!-- ============================================================================================ -->

     <!-- SML-GetCosem                                                                          -->
     <!-- ..................................................................................... -->

<xs:complexType         name = "SML_GetCosem.Req">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"
                                                      minOccurs = "0" />
        <xs:element     name = "username"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "password"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "objName"              type      = "xs:hexBinary"/>
        <xs:element     name = "classId"              type      = "xs:short"/>
        <xs:element     name = "classVersion"         type      = "xs:short"/>
        <xs:element     name = "attributeIndexList"   type      = "SML_CosemAttrIndexList"
                                                      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_GetCosem.Res">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"/>
        <xs:element     name = "objName"              type      = "xs:hexBinary"/>
        <xs:element     name = "classId"              type      = "xs:short"/>
        <xs:element     name = "classVersion"         type      = "xs:short"/>
        <xs:element     name = "attributeList"        type      = "SML_CosemAttrList"/>
    </xs:sequence>
</xs:complexType>


     <!-- SML-SetCosem                                                                          -->
     <!-- ..................................................................................... -->

<xs:complexType         name = "SML_SetCosem.Req">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"
                                                      minOccurs = "0" />
        <xs:element     name = "username"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "password"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "objName"              type      = "xs:hexBinary"/>
        <xs:element     name = "classId"              type      = "xs:short"/>
        <xs:element     name = "classVersion"         type      = "xs:short"/>
        <xs:element     name = "attributeList"        type      = "SML_CosemAttrList"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_SetCosem.Res">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"/>
        <xs:element     name = "objName"              type      = "xs:hexBinary"/>
        <xs:element     name = "classId"              type      = "xs:short"/>
        <xs:element     name = "classVersion"         type      = "xs:short"/>
        <xs:element     name = "attributeList"        type      = "SML_CosemAttrList"
                                                      minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>


     <!-- SML-RunCosem                                                                          -->
     <!-- ..................................................................................... -->

<xs:complexType         name = "SML_RunCosem.Req">
    <xs:sequence>
        <xs:element     name = "clientId"             type      = "xs:hexBinary"/>
        <xs:element     name = "serverId"             type      = "xs:hexBinary"
                                                      minOccurs = "0" />
        <xs:element     name = "username"             type      = "xs:hexBinary"
                                                      minOccurs = "0"/>
```

```
        <xs:element     name = "password"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "objName"             type      = "xs:hexBinary"/>
        <xs:element     name = "classId"             type      = "xs:short"/>
        <xs:element     name = "classVersion"        type      = "xs:short"/>
        <xs:element     name = "serviceIndex"        type      = "xs:byte"/>
        <xs:element     name = "attributeList"       type      = "SML_CosemAttrList"
                                                     minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name = "SML_RunCosem.Res">
    <xs:sequence>
        <xs:element     name = "clientId"            type      = "xs:hexBinary"
                                                     minOccurs = "0"/>
        <xs:element     name = "serverId"            type      = "xs:hexBinary"/>
        <xs:element     name = "objName"             type      = "xs:hexBinary"/>
        <xs:element     name = "classId"             type      = "xs:short"/>
        <xs:element     name = "classVersion"        type      = "xs:short"/>
        <xs:element     name = "attributeList"       type      = "SML_CosemAttrList"
                                                     minOccurs = "0"/>
    </xs:sequence>
</xs:complexType>


     <!--COSEM attributes                                                                    -->
     <!-- ..................................................................................... -->

<xs:simpleType          name      = "SML_CosemAttrIndexList">
    <xs:list            itemType  = "xs:short"/>
</xs:simpleType>

<xs:complexType         name      = "SML_CosemAttrList">
    <xs:sequence>
        <xs:element     name      = "SML_CosemAttr"   minOccurs = "0"     maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name      = "SML_CosemAttr">
    <xs:sequence>
        <xs:element     name      = "attributeIndex"       type = "xs:short"/>
        <xs:element     name      = "attributeContent"     type = "SML_CosemValue"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name      = "SML_CosemValue">
    <xs:choice>
        <xs:element     name      = "valBoolean"           type = "xs:boolean"/>
        <xs:element     name      = "valByteList"          type = "xs:hexBinary"/>
        <xs:element     name      = "valInteger8"          type = "xs:byte"/>
        <xs:element     name      = "valInteger16"         type = "xs:short"/>
        <xs:element     name      = "valInteger32"         type = "xs:int"/>
        <xs:element     name      = "valInteger64"         type = "xs:long"/>
        <xs:element     name      = "valUnsigned8"         type = "xs:unsignedByte"/>
        <xs:element     name      = "valUnsigned16"        type = "xs:unsignedShort"/>
        <xs:element     name      = "valUnsigned32"        type = "xs:unsignedInt"/>
        <xs:element     name      = "valUnsigned64"        type = "xs:unsignedLong"/>
        <xs:element     name      = "valList"              type = "SML_CosemList"/>
        <xs:element     name      = "valStruct"            type = "SML_CosemList"/>
        <xs:element     name      = "valArray"             type = "SML_CosemList"/>
    </xs:choice>
</xs:complexType>

<xs:complexType         name      = "SML_CosemList">
    <xs:sequence>
        <xs:element     name      = "SML_CosemListEntry"   minOccurs = "0"
                                                           maxOccurs = "unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType         name      = "SML_CosemListEntry">
    <xs:choice>
        <xs:element     name      = "valBoolean"           type = "xs:boolean"/>
        <xs:element     name      = "valByteList"          type = "xs:hexBinary"/>
        <xs:element     name      = "valInteger8"          type = "xs:byte"/>
        <xs:element     name      = "valInteger16"         type = "xs:short"/>
        <xs:element     name      = "valInteger32"         type = "xs:int"/>
        <xs:element     name      = "valInteger64"         type = "xs:long"/>
        <xs:element     name      = "valUnsigned8"         type = "xs:unsignedByte"/>
        <xs:element     name      = "valUnsigned16"        type = "xs:unsignedShort"/>
        <xs:element     name      = "valUnsigned32"        type = "xs:unsignedInt"/>
        <xs:element     name      = "valUnsigned64"        type = "xs:unsignedLong"/>
    </xs:choice>
</xs:complexType>


</xs:schema>
```

Fig. 4: XML scheme file for coding of SML per XML

Annex A (normative)

A.1 SML Messages for non COSEM environments

A.1.1 SML_GetProfilePack.Req

Deprecated: not to be used in a COSEM environment

In an SML order file one or more SML_GetProfilePack messages can be present.

Each SML_GetProfilePack.Req serves to request individual measured values or measured-value lists, which are transmitted in packed form.

The client must respond to each SML_GetProfilePack.Req with precisely one SML_GetProfilePack.Res or one SML_Attention.Res. The empty 'ServerID' must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(RR)  SML_GetProfilePack.Req   ::=   SEQUENCE
      {
        serverId                      Octet String OPTIONAL,
        username                      Octet String OPTIONAL,
        password                      Octet String OPTIONAL,
        withRawdata                   Boolean OPTIONAL,
        beginTime                     SML_Time OPTIONAL,
        endTime                       SML_Time OPTIONAL,
        parameterTreePath             SML_TreePath,
        object_List                   List_of_SML_ObjReqEntry OPTIONAL,
        dasDetails                    SML_Tree OPTIONAL
      }
(SS)  List_of_SML_ObjReqEntry  ::=   SEQUENCE OF
      {
        object_List_Entry             SML_ObjReqEntry
      }
(TT)  SML_ObjReqEntry          ::=   Octet String
```

The 'parameterTreePath' element designates the measured-value list and thus serves for classification into variants like logbook, load profile or comparable. It must be occupied with at least one Code Number {for coding, see Paragraphs (LLL) and 0}.

The 'object_List' is required to enable the desired channel for reading off load profiles to be stated with the order. If this element is not stated or empty, then all channels included in the profile must be stated in the response.

The desired target range is stated using time limits.

The following agreement shall apply if time limits are defined per seconds index[5]:

The seconds index with the value ‘00000000’ always describes the value lying temporally furthest in the past.

The seconds index with the value ‘FFFFFFFF’ always describes the ‘current entry’, meaning the value lying temporally directly before the ongoing time.

All values between ‘00000000’ and ‘FFFFFFFF’ are offsets to be counted from the beginning (‘00000000’).

If in the case of range queries the desired seconds index is not available, then it must be used in the response:

... if present for the beginning of a range,
    always the value lying temporally directly afterwards[6] or
  ... if this is not present,
    the ‘current entry’

... if present for the end of a range,
    always the value lying temporally before it[7] or
  ... if this is not present,
    the ‘current entry’

.

If ‘beginTime’ is not stated, the value ‘00000000’ must be assumed.

If ‘endTime’ is not stated, the value ‘FFFFFFFF’ must be assumed.

Per ‘dasDetails’ parameters required as supplement for data procurement can be transmitted at need. These are transmitted as a tree structure - see Section A.1.6.

A.1.2 SML_GetProfilePack.Res

Deprecated: not to be used in a COSEM environment

In an SML response file, one or more SML_GetProfilePack.Res messages can be present.

Each SML data source supplies precisely one SML_GetProfilePack.Res response to one SML_GetProfilePack.Req request.

---

[5] The duty environment at present provides solely for use of the seconds index; see the explanation on ‘SML_Time’. Possible future expansion may designate further alternatives for this.

[6] Towards larger indices.

[7] Towards smaller indices.

```
(UU)  SML_GetProfilePack.Res    ::=     SEQUENCE
      {
        serverId                        Octet String,
        actTime                         SML_Time,
        regPeriod                       Unsigned32,
        parameterTreePath               SML_TreePath,
        header_List                     List_of_SML_ProfObjHeaderEntry,
        period_List                     List_of_SML_ProfObjPeriodEntry,
        rawdata                         Octet String OPTIONAL,
        profileSignature                SML_Signature OPTIONAL
      }

(VV)  List_of_SML_ProfObjHeaderEntry    ::=  SEQUENCE OF
      {
        header_List_Entry               SML_ProfObjHeaderEntry
      }

(WW)  SML_ProfObjHeaderEntry    ::=     SEQUENCE
      {
        objName                         Octet String,
        unit                            SML_Unit,
        scaler                          Integer8
      }

(XX)  List_of_SML_ProfObjPeriodEntry    ::=  SEQUENCE OF
      {
        period_List_Entry               SML_ProfObjPeriodEntry
      }

(YY)  SML_ProfObjPeriodEntry    ::=     SEQUENCE
      {
        valTime                         SML_Time,
        status                          Unsigned64,
        value_List                      List_of_SML_ValueEntry,
        periodSignature                 SML_Signature OPTIONAL
      }

(ZZ)  List_of_SML_ValueEntry    ::=     SEQUENCE OF
      {
        value_List_Entry                SML_ValueEntry
      }

(AAA) SML_ValueEntry            ::= SEQUENCE
      {
        value                           SML_Value,
        valueSignature                  SML_Signature OPTIONAL
      }
```

(BBB) SML_Value ::= IMPLICIT CHOICE
{
boolean-Value boolean,
byte-List Octet String[8],
8-Bit-Integer Integer8,
16-Bit-Integer Integer16,
32-Bit-Integer Integer32,
64-Bit-Integer Integer64,
8-Bit-Unsigned Unsigned8,
16-Bit-Unsigned Unsigned16,
32-Bit-Unsigned Unsigned32,
64-Bit-Unsigned Unsigned64
smlList SML_ListType
}

(CCC) SML_Unit ::= Unsigned8

For numerical values, see DLMS-Unit list, to be found, for example, in IEC 62056-62.

The number of elements in the 'header_List' must always be identical to the number of elements in the list of registration periods ('value_List').

The arrangement of registration periods in the list is based on the pattern "the value with the smallest second index comes first".

The 'actTime' element supplies the time information that was present at the data source at the moment when execution of the order began.

The 'valTime' element supplies the time information that was present at the data source sat the moment when the measured value was formed.

The 'regPeriod' element specifies the duration of the registration period being used. The value is stated in seconds. If the recording involved is an event-oriented profile, (i.e. no specific registration period is on file), '0' must be used as the value.

Use of the two mutually independent lists ('header_List' and 'period_List') has been chosen to be able to omit multiple acquisition of redundant information, such as seconds index and status at the moment of measured-value formation for each registration period in case of several load profiles ("+A", "-A", ...).

Use of either 'profileSignature', 'periodSignature' or 'valueSignature' enables entire load profiles, individual measured values and also tuples of entire registration periods to be jointly protected. Which approach is adopted will depend on the application involved.

The 'scaler' field is used to establish the relationship between the unit and the numerical value as follows:

$$\text{numerical value} = \text{SML\_Value} \times 10^{\text{scaler}}$$

[8] Note: It is allowed to use an Octet String of length '0'.

Per ‚smlList' in an SML_Value the following data structures can be transported:

```
(DDD)  SML_ListType              ::= CHOICE
       {
         smlTime                 [0x01]   SML_Time
         smlTimestampedValue     [0x02]   SML_TimestampedValue
       }

(EEE)  SML_TimestampedValue      ::= SEQUENCE
       {
         smlTime                          SML_Time
         status                           SML_Status
         simpleValue                      SML_SimpleValue
       }

(FFF)  SML_SimpleValue           ::= IMPLICIT CHOICE
       {
         boolean-Value                    boolean,
         byte-List                        Octet String[8],
         8-Bit-Integer                    Integer8,
         16-Bit-Integer                   Integer16,
         32-Bit-Integer                   Integer32,
         64-Bit-Integer                   Integer64,
         8-Bit-Unsigned                   Unsigned8,
         16-Bit-Unsigned                  Unsigned16,
         32-Bit-Unsigned                  Unsigned32,
         64-Bit-Unsigned                  Unsigned64
       }
```

A.1.3 SML_GetProfileList.Req

Deprecated: not to be used in a COSEM environment

In an SML order file, one or more SML_GetProfileList messages can be present.

Each SML_GetProfileList.Req serves to request individual measured values or lists of measured values, which are transmitted in simple list form. In contrast to SML_GetProfilePack, where the responses are as far as possible transmitted without redundancy and optimally packed, SML_GetProfile expects simple lists. These offer the disadvantage of generating substantially more volume when transmitting daily load profiles, but in return provide the advantage of a simple structure, which becomes significantly more efficient in use with systems featuring quarter-hourly data procurement.

The client can respond to each SML_GetProfileList.Req with one or more SML_GetProfileList.Res or one SML_Attention.Res. The empty 'ServerID' must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(GGG)  SML_GetProfileList.Req      ::=    SEQUENCE
       {
         serverId                         Octet String OPTIONAL,
         username                         Octet String OPTIONAL,
         password                         Octet String OPTIONAL,
         withRawdata                      Boolean OPTIONAL,
         beginTime                        SML_Time OPTIONAL,
         endTime                          SML_Time OPTIONAL,
         parameterTreePath                SML_TreePath,
         object_List                      List_of_SML_ObjReqEntry OPTIONAL,
         dasDetails                       SML_Tree OPTIONAL
       }
```

For the elements relating to SML_GetProfileList.Req, the stipulations defined under SML_GetProfilePack.Req apply.

A.1.4 SML_GetProfileList.Res

Deprecated: not to be used in a COSEM environment

In an SML response file, one or more SML_GetProfileList.Res messages can be present.

Each SML data source supplies one or more SML_GetProfileList.Res responses to an SML_GetProfileList.Req order. Each SML_GetProfileList.Res response contains the measured value of a registration period (thus usually of a quarter of an hour) in the sense of an atom.

```
(HHH)  SML_GetProfileList.Res      ::=    SEQUENCE
       {
         serverId                         Octet String,
         actTime                          SML_Time,
         regPeriod                        Unsigned32,
         parameterTreePath                SML_TreePath,
         valTime                          SML_Time,
         status                           Unsigned64,
         period_List                      List_of_SML_PeriodEntry,
         rawdata                          Octet String OPTIONAL,
         periodSignature                  SML_Signature OPTIONAL
       }

(III)  List_of_SML_PeriodEntry     ::=    SEQUENCE OF
       {
         period_List_Entry                SML_PeriodEntry
       }
```

```
(JJJ)   SML_PeriodEntry          ::= SEQUENCE
        {
          objName                    Octet String,
          unit                       SML_Unit,
          scaler                     Integer8,
          value                      SML_Value,
          valueSignature             SML_Signature OPTIONAL
        }
```

Use of either ‘periodSignature’ or ‘valueSignature’ enables both individual measured values and also tuples of entire registration periods to be jointly protected. Which approach is adopted will depend on the application involved.

A.1.5 SML_GetProcParameter.Req

Deprecated: not to be used in a COSEM environment

SML_GetProcParameter.Req enable operating parameters (modem parameters, protocol parameters, capacity utilisation of software modules, ...) to be retrieved in an SML order file.

The client responds to this message per SML_GetProcParameter.Res or SML_Attention.Res in the response file. The empty ‘ServerID’ must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

An individual parameter is always defined by its OBIS Code Number. It is from the OBIS Code Number that the specific data type is implicitly derived, and the value range for the result to be supplied. The specific assignment and permissible OBIS Code Numbers and their significance can be taken for the appendix. Within the coding of an SML message, the data types are also coded, so that the generic implementation for converting the parameters into internal programs is easily possible.

Since the structure and number of the parameters required for a specific application case is unknown beforehand, but continued modification of the SML specification to new requirements is to be avoided if possible, the ‘SML_..._ProcParameter’ messages use a tree structure, so that parameter trees of any desired configuration can be transported.

```
(KKK)   SML_GetProcParameter.Req   ::=  SEQUENCE
        {
          serverId                   Octet String OPTIONAL,
          username                   Octet String OPTIONAL,
          password                   Octet String OPTIONAL,
          parameterTreePath          SML_TreePath
          attribute                  Octet String OPTIONAL,
        }

(LLL)   SML_TreePath               ::=     SEQUENCE OF
        {
          path_Entry                 Octet String
        }
```

SML_TreePath enables parameters to be selectively addressed within a parameter tree. Note that an SML_TreePath always begins from the root element of the parameter tree. The meaning (corresponding to the address) of the root element is given with its parameter name.

Thanks to the list structure, the parameter name of the root element can always be found in that SML_TreePath element that can be found as the first and thus directly under 'SML_GetProcParameter.Req'.

The 'attribute' characteristic can at need be also provided with a restriction/detailed information for narrowing the desired request result.

A.1.6 SML_GetProcParameter.Res

Deprecated: not to be used in a COSEM environment

SML_GetProcParameter.Res is used in an SML response file for acknowledging the order SML_GetProcParameter.Req.

```
(MMM)  SML_GetProcParameter.Res      ::=  SEQUENCE
       {
         serverId                        Octet String,
         parameterTreePath               SML_TreePath,
         parameterTree                   SML_Tree
       }

(NNN)  SML_Tree                 ::=     SEQUENCE
       {
         parameterName                   Octet String,
         parameterValue                  SML_ProcParValue OPTIONAL,
         child_List                      List_of_SML_Tree OPTIONAL
       }

(OOO)  SML_ProcParValue         ::= CHOICE
       {
         smlValue                [0x01]  SML_Value[9],
         smlPeriodEntry          [0x02]  SML_PeriodEntry,
         smlTupelEntry           [0x03]  SML_TupelEntry,
         smlTime                 [0x04]  SML_Time
         smlListEntry            [0x05]  SML_ListEntry
       }
```

[9] Note: Applications shall define if contents of the type Typ SML_Time shall be transmitted directly per choice ‚0x04' or indirectly per content of SML_Value.

```
(PPP)  SML_TupelEntry          ::= SEQUENCE
       {
         serverId              Octet String,
         secIndex              SML_Time,
         status                Unsigned64,
         unit_pA               SML_Unit,
         scaler_pA             Integer8,
         value_pA              Integer64,
         unit_R1               SML_Unit,
         scaler_R1             Integer8,
         value_R1              Integer64,
         unit_R4               SML_Unit,
         scaler_R4             Integer8,
         value_R4              Integer64,
         signature_pA_R1_R4    Octet String,
         unit_mA               SML_Unit,
         scaler_mA             Integer8,
         value_mA              Integer64,
         unit_R2               SML_Unit,
         scaler_R2             Integer8,
         value_R2              Integer64,
         unit_R3               SML_Unit,
         scaler_R3             Integer8,
         value_R3              Integer64,
         signature_mA_R2_R3    Octet String
       }

(QQQ)  List_of_SML_Tree        ::=    SEQUENCE OF
       {
         tree_Entry                   SML_Tree
       }
```

‘SML_Tree’ enables individual parameters (leaves or nodes) to be built up with their (in the case of nodes) following children (see ‘child_List’). Specifically, this means an ‘SML_Tree’ can thus be used to image …
… an individual parameter,
… a node with a dependent list of further parameters or
… a node with a dependent list of further subtrees.

The ‘parameterValue’ and ‘child_List’ elements are both optional, so that the following variants can be imaged:

– a parameter consisting of name (coded per OBIS Code Number) and value;
– a parameter consisting of name (coded per OBIS Code Number) and subtree or list of further parameters
– a parameter consisting of name (coded per OBIS Code Number), value and subtree or list of further parameters.

Whether an ‘SML_Tree’ is to be evaluated as a leaf or node is specified by the presence of the ‘child_List’ element. If an ‘SML_Tree’ has the ‘child_List’ element, it is a node; if not, it is a leaf.

A.1.7 SML_SetProcParameter.Req

Deprecated: not to be used in a COSEM environment

SML_SetProcParameter.Req enables operating parameters in an SML order file (for example, the baudrate for accessing a terminal) to be transmitted.

For structuring, please consult the notes in Section A.1.5.

```
(RRR)  SML_SetProcParameter.Req      ::=  SEQUENCE
       {
         serverId                    Octet String OPTIONAL,
         username                    Octet String OPTIONAL,
         password                    Octet String OPTIONAL,
         parameterTreePath           SML_TreePath,
         parameterTree               SML_Tree
       }
```

The client responds to this message per SML_Attention.Res. The empty ‘ServerID’ must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

A.1.8 SML_GetList.Req

Deprecated: not to be used in a COSEM environment

Per SML_GetList.Req a parameterized data list laid down in the server can be retrieved. As an answer an SML_GetList.Res or an SML_Attention.Res must be generated. The empty ‘ServerID’ must be considered as broadcast address for the messages which have to be answered. If an application in contrast to this uses special broadcast or multitask addresses these have to be defined explicitly in the application together with the behaviour.

```
(SSS)  SML_GetList.Req          ::=     SEQUENCE
       {
         clientId                       Octet String,
         serverId                       Octet StringOPTIONAL,
         username                       Octet StringOPTIONAL,
         password                       Octet StringOPTIONAL,
         listName                       Octet StringOPTIONAL
       }
```

Per attribute ‚listName’ the desired list can be named. It defines per OBIS the desired quantity / list. Since this element is denoted as „optional“ simple devices can omit its interpretation and send only the meter reading whose meaning they have to give per OBIS code in the answer.

A.1.9 SML_GetList.Res

Deprecated: not to be used in a COSEM environment

Per SML_GetList.Res a list of predefined data can be transmitted.

```
(TTT)  SML_GetList.Res        ::=  SEQUENCE
       {
         clientId                  Octet String  OPTIONAL,
         serverId                  Octet String,
         listName                  Octet String  OPTIONAL,
         actSensorTime             SML_Time      OPTIONAL,
         valList                   SML_List,
         listSignature             SML_Signature OPTIONAL,
         actGatewayTime            SML_Time      OPTIONAL
       }
```

With the optional attribute ‚actSensorTime' a sensor which generates this SML message can attach its own current time information.

With the optional attribute ‚actGatewayTime' a gateway which transports this SML message can attach its own current time information.

```
(UUU)  SML_List               ::=  SEQUENCE OF
       {
       valListEntry               SML_ListEntry
       }
(VVV)  SML_ListEntry          ::=  SEQUENCE
       {
         objName                  Octet String,
         status                   SML_Status    OPTIONAL,
         valTime                  SML_Time      OPTIONAL,
         unit                     SML_Unit      OPTIONAL,
         scaler                   Integer8      OPTIONAL,
         value                    SML_SimpleValue,
         valueSignature           SML_Signature OPTIONAL
       }
(WWW)  SML_Status             ::=  IMPLICIT CHOICE
       {
         status8                  Unsigned8
         status16                 Unsigned16
         status32                 Unsigned32
         status64                 Unsigned64
       }
```

Annex B (informative)

B.1 SML Transport Protocol (CHAPTER has to move out …)

B.1.1 Version 1

For transmitting SML messages over unsecured links, such as direct optical readout per MED on the spot, readout via PSTN modem, GSM modem or comparable routes, the SML Transport Protocol is defined.

It can also be used with secure links (such as TCP).

As a streaming protocol, it follows the approach of coding the data traffic “on the fly” in the terminals, thus enabling it to dispense with the use of extensive buffers.

The rules are is defined as follows:

– Beginning, end and other features of a message are identified using escape sequences.

– An escape sequence is initiated with an escape string. This escape string is followed by the feature added to the message (beginning, end, …).

– If the escape string occurs at any position in the payload stream itself, this will itself be transmitted as an escape sequence. In this case, the feature added to the message is the escape string (it is thus transmitted twice in succession).

– The number of bytes in the payload stream is always extended at the end to a quantity divisible by the number of bytes in the escape string without leaving a remainder[10]. For this extension, bytes with the content ‘00’ (hex) are used.

As an escape string, the byte string ‘1b 1b 1b 1b’ (stated in hex.) is specified.

The following escape features have been defined:

Tab. 7: Escape features for the SML Transport Protocol

Pos.	Escape feature (stated in hex)	Significance / Note
1	1b 1b 1b 1b	Designates the case that the escape sequence is itself contained in the payload stream.
2	01 01 01 01	Initiates a transmission of version 1 as data flow. Designates the feature ‘Beginning of a message’.
3	02 TT UU VV	Initiates a transmission of version 2 as block transfer, see chapter B.1.2.
4	03 00 RR RR	Used only in connection with version 2, defines the timeout to be used - see chapter B.1.2.
5	04 00 SS SS	Used only in connection with version 2, defines the block size to be used - see chapter B.1.2.
6	1a XX YY ZZ	Designates the feature “End of a message’. ‘XX’ ⇔ Can assume values in the range ‘00’, ‘01’, ‘02’ or ‘03’ and supplies the number of bytes that have been added at the end of the useful load to enable the number of bytes in the useful load to be divided by the number of bytes in an escape string without

[10] Since the escape string consists of 4 bytes, the number of bytes in the useful load must always likewise be increased in such a way that a division modulo 4 supplies precisely 0. This means the sender must add either no byte, or one, two or three bytes.

Pos.	Escape feature (stated in hex)	Significance / Note
		any remainder. 'YY ZZ' ⇔ Can assume values in the range '00..FF' and contains the checksum for the entire message. YY is here the Most Significant Byte and ZZ the Least Significant Byte of the checksum.
7	All other combinations	Reserved for future expansions.

The checksum must be calculated in accordance with CCITT-CRC16. It is calculated using all bytes of the data flow in the SML Transport Protocol with the exception of the last two bytes (and thus without the bytes of the checksum itself).

Calculation is performed in accordance with IEC 62056-46.

If data procurement is performed over TCP or UDP links, the SML Transport Protocol is used for identifying self-contained SML files.

B.1.2 Version 2

For the SML file transmission per unsecured half-duplex connections the SML transport protocol of version 2 can be used. This yields a simple mechanism for flow control so allowing the sender the adaptation of his outputs to the requirements of the clients having possibly only small resources.

The SML transport protocol version 2 therefore also must be used if SML files are transferred per optical ways comparable to IEC 62056-21.

The concept which was realised to fulfil these requirements divides the SML file to be transferred into blocks and uses timeouts for the restart after errors. The block sizes as well the timeouts are negotiated between sender and receiver at the beginning of the transmission. As timeout value and as minimum block size are used:

Initially to use timeout: 5 s;
Initially to use block size: 32 Bytes.

Compared with version 1 the version 2 uses three additionally defined ESC sequences.

B.1.3 Initiating of the transmission in version 2

To distinguish version 2 of the SML transport protocol from version 1 (see chapter B.1.1) here the sequence

1b 1b 1b 1b 02 TT UU VV

is used as start sequence (all data in hex, see Tab. 7).

The elements „TT UU VV“ serve as labelling of blocks and SML files.

B.1.4 Labelling of blocks

Blocks are labelled per element „TT“, see Tab. 7. This labelling uses the following features:

– Bit 7, MSB: 0 ⇔ sent block,
1 ⇔ received as ACK;

– Bit 6: 0 ⇔ further blocks following,
1 ⇔ last block of the SML file.

– Bit 5 … Bit 0, LSB: Block number, starting with 0x00, cyclically stepping to 0x01 when 0x3F is reached.

B.1.5 Labelling of SML files

SML files are labelled per element „UU“, see Tab. 7. This labelling uses the following features:

– The SML file which must be sent first over the transmission line gets the labelling ‚0x00’;

– With each of the following transmitted SML files the feature is increased by one;

– When the value ‚0xFF’ is reached the next one starts with ‚0x00’ again.

B.1.6 Feature ‚VV’

The element „VV“, see Tab. 7, is reserved for future extensions and must be set always equal to ‚0x01’.

B.1.7 Negotiation of the timeout to be used

At the beginning of a transmission the sender proposes the timeout to be used. For this an ESC sequence of the form

1b 1b 1b 1b 03 00 RR RR

is used where in „RR RR“ the proposed timeout in ‘ms’ is given.

The timeout must be noted in the ESC sequence as

1b 1b 1b 1b 03 00 high byte low byte.

The receiver answers to this ESC sequence with the timeout which has been chosen by it where only the same or a higher value is allowed. The sender must use the timeout defined by the receiver.

B.1.8 Negotiation of the maximum allowed block size

At the beginning of a transmission the sender proposes the maximum block size to be used. For this an ESC sequence of the form

1b 1b 1b 1b 04 00 SS SS

is used where in „SS SS“ the proposed timeout in ‘bytes’ is given.

The block size must be noted in the ESC sequence as

1b 1b 1b 1b 04 00 high byte low byte.

The receiver answers to this ESC sequence with the block size which has been chosen by it where only the same or a lower value is allowed. The sender must use the block size defined by the receiver.

B.1.9 Process of establishing a transmission

A version 2 transmission is initiated by the following process:

The sender sends the first data block which has to be exactly of the following form:
1b 1b 1b 1b 02 00 UU VV
1b 1b 1b 1b 03 00 RR RR (with RR RR proposed timeout in ms)
1b 1b 1b 1b 04 00 SS SS (with SS SS proposed block size in Byte)
1b 1b 1b 1b 1a xx yy zz

The receiver acknowledges this first data block with:
1b 1b 1b 1b 02 80 UU VV
1b 1b 1b 1b 03 00 rr rr (with rr rr confirmed timeout in ms)

1b 1b 1b 1b 04 00 ss ss (with ss ss confirmed block size in Byte)
1b 1b 1b 1b 1a xx yy zz

B.1.10 Process of the course of a transmission

A version 2 transmission is initiated as described above and then runs as a „ping-pong“ of sent and acknowledging data blocks.

The finite state machine after a single timeout falls back to the beginning of the current transmission step if it gets no acknowledge for the block last sent.

In case of error after a double timeout the complete transmission must be repeated.

B.1.11 Example of the course of a version 2 transmission process

As an example for the transmission process the following course is given:

– Start of transmission with:
  1b 1b 1b 1b 02 00 UU VV
  1b 1b 1b 1b 03 00 RR RR (with RR RR proposed timeout in ms)
  1b 1b 1b 1b 04 00 SS SS (with SS SS proposed block size in Byte)
  1b 1b 1b 1b 1a xx yy zz
– If the start of transmission is not correctly decoded /received, no reaction is shown by the receiver. The sender must initiate the transmission once more.
– Transmission confirmed by:
  1b 1b 1b 1b 02 80 UU VV
  1b 1b 1b 1b 03 00 rr rr (with rr rr confirmed timeout in ms)
  1b 1b 1b 1b 04 00 ss ss (with ss ss confirmed block size in Byte)
  1b 1b 1b 1b 1a xx yy zz
– First payload is sent with:
  1b 1b 1b 1b 02 01 UU VV
  payload
  1b 1b 1b 1b 1a xx yy zz
– First payload acknowledged with ACK:
  1b 1b 1b 1b 02 81 UU VV
  1b 1b 1b 1b 1a xx yy zz
– First payload is rejected with NAK:
  1b 1b 1b 1b 02 80 UU VV (with „old“ block number in 0x80)
  1b 1b 1b 1b 1a xx yy zz
– Second payload is sent with:
  1b 1b 1b 1b 02 02 UU VV
  payload
  1b 1b 1b 1b 1a xx yy zz
– Second payload acknowledged with ACK:
  1b 1b 1b 1b 02 82 UU VV
  1b 1b 1b 1b 1a xx yy zz
– Second payload is rejected with NAK:
  1b 1b 1b 1b 02 81 UU VV (with „old“ block number in 0x81)
  1b 1b 1b 1b 1a xx yy zz
– Third and last payload is sent with:
  1b 1b 1b 1b 02 43 UU VV
  payload
  1b 1b 1b 1b 1a xx yy zz
– Third and last payload acknowledged with ACK:
  1b 1b 1b 1b 02 C3 UU VV
  1b 1b 1b 1b 1a xx yy zz

– Third and last payload rejected with NAK:
1b 1b 1b 1b 02 82 UU VV (with „old“ block number in 0x81)
1b 1b 1b 1b 1a xx yy zz