

	DIN VDE 0418-63-9 (VDE 0418-63-9)	DIN
	Diese Norm ist zugleich eine VDE-Bestimmung im Sinne von VDE 0022. Sie ist nach Durchführung des vom VDE-Präsidium beschlossenen Genehmigungsverfahrens unter der oben angeführten Nummer in das VDE-Vorschriftenwerk aufgenommen und in der „etz Elektrotechnik + Automation“ bekannt gegeben worden.	VDE
ICS 35.240.99; 91.140.50 Messeinrichtungen und -systeme – Teil 63-9: Intelligentes Kommunikationsprotokoll für Elektrizitätszähler (SML) Metering equipment and systems – Part 63-9: Smart Message Language for electricity meters (SML) Équipements et systèmes de comptage – Partie 63-9: Intelligent protocole pour les compteurs d'électricité (SML) Gesamtumfang 47 Seiten DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE		

Anwendungsbeginn

Anwendungsbeginn dieser Norm ist 2022-02-01.

	Inhalt	Seite
Vorwort.....		5
1	Anwendungsbereich	6
2	Normative Verweisungen	6
3	Begriffe und Abkürzungen	7
3.1	Begriffe	7
3.2	Abkürzungen	7
4	Smart Message Language (SML)	8
4.1	Struktur	8
4.2	SML-Datei.....	8
4.3	Grundaufbau von SML-Nachrichten.....	9
5	SML-Nachrichten.....	10
5.1	Allgemeines	10
5.2	SML-Bezeichner	12
5.2.1	SML_PublicOpen.Req.....	12
5.2.2	SML_PublicOpen.Res	13
5.2.3	SML_PublicClose.Req.....	15
5.2.4	SML_PublicClose.Res.....	15
5.2.5	SML_GetProfilePack.Req.....	15
5.2.6	SML_GetProfilePack.Res.....	17
5.2.7	SML_GetProfileList.Req	20
5.2.8	SML_GetProfileList.Res	20
5.2.9	SML_GetProcParameter.Req	21
5.2.10	SML_GetProcParameter.Res.....	22
5.2.11	SML_SetProcParameter.Req.....	24
5.2.12	SML_Attention.Res.....	24
5.2.13	Globale SML-Attentionnummer	25
5.2.14	SML_GetList.Req	26
5.2.15	SML_GetList.Res	27
5.3	Zugriff auf COSEM-Objekt per SML.....	28
5.4	Application-Container	31
5.4.1	Allgemeines	31
5.4.2	Application-Container zur Inhaltsdatenverschlüsselung	31
5.4.3	Application-Container zur Ablösung von ‚SML_TupelEntry‘.....	33
6	SML binary encoding, direkt gepackte Kodierung.....	36
6.1	Allgemeines	36

6.2	Type-Length-Field	36
6.3	Kodierung der Datentypen	37
6.3.1	Datentyp Octet String	37
6.3.2	Datentypen Integer8, Integer16, Integer32 und Integer64	37
6.3.3	Datentypen Unsigned8, Unsigned16, Unsigned32 und Unsigned64	38
6.3.4	Datentyp Boolean	39
6.3.5	Datentyp List of	39
6.4	Kodierung besonderer Merkmale	39
6.4.1	Merkmal Ende einer SML-Nachricht	39
6.4.2	Merkmal SEQUENCE	40
6.4.3	Merkmal CHOICE	40
6.4.4	Merkmal OPTIONAL	40
7	XML-Kodierung	40
8	SML-Transport-Protokoll	40
8.1	Version 1	40
8.2	Version 2	41
8.2.1	Allgemeines	41
8.2.2	Übertragung nach Version 2	42
8.2.3	Vereinbarung des zu verwendenden Timeouts	42
8.2.4	Vereinbarung der maximal zulässigen Blocksize	43
8.2.5	Prozess zum Aufbau der Übertragung	43
8.2.6	Prozess zum Ablauf einer Übertragung	43
8.2.7	Beispiel für einen Ablauf des Übertragungsvorgangs nach Version 2	43
9	Benötigte Kennzahlen	44
Anhang A (normativ)	XML-Schema-Definition (XSD) SML	46
Literaturhinweise		47

Bilder

Bild 1 – Einordnung von SML in das Umfeld europäischer / internationaler Normen	6
Bild 2 – SML-Nachrichten und Kommunikationswege	8
Bild 3 – SML-Kommunikationsmodell	9
Bild 4 – COSEM-Objekt-Attribut-Referenz(en) als Baum von Byte-Ketten im ‚SML_TreePath‘	28
Bild 5 – Attribut-Inhalt(e) als Baum von Byte-Ketten im ‚SML_Tree‘	29
Bild 6 – COSEM-Objekt-Methoden-Referenz als Baum von Byte-Ketten in einem ‚SML_Tree‘	31
Bild 7 – Anwendungshinweis zum Einsatz der Inhaltsdatenverschlüsselung	33

Tabellen

Tabelle 1 – Beispiel zur Verwendung des Merkmals ‚Gruppen-Nummer‘	11
Tabelle 2 – Liste globaler Fehlernummern	25
Tabelle 3 – Liste globaler Hinweisnummern	26
Tabelle 4 – Zuordnung von COSEM- und SML-Datentypen	29
Tabelle 5 – Application-Container zur Inhaltsdatenverschlüsselung	32

Tabelle 6 – Application-Container zur Ablösung von ‚SML_TupelEntry‘	34
Tabelle 7 – Bitkodierung im Type-Length-Field für das erste Byte einer TL-Field-Angabe	36
Tabelle 8 – Bitkodierung im Type-Length-Field für das zweite und folgende TL-Field-Bytes	37
Tabelle 9 – Bitkodierung im Type-Length-Field für einen Octet String.....	37
Tabelle 10 – Escape-Merkmale zum SML-Transport-Protokoll.....	41
Tabelle 11 – Kennzahlen-Liste zu den ‚Application Container‘-Datenstrukturen	45

Vorwort

Vorausgegangener Norm-Entwurf: E DIN 43863-9:2018-11.

Für dieses Dokument ist das nationale Arbeitsgremium DKE/K 461 „Messeinrichtungen und -systeme für Elektrizität“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (www.dke.de) zuständig.

Aktuelle Informationen zu diesem Dokument können über die Internetseiten von DKE (www.dke.de) und DIN (www.din.de) durch eine Suche nach der Dokumentennummer aufgerufen werden.

Zu diesem Dokument gehört eine Beilage, die Sie unter www.vde-normen.de/0400493.zip herunterladen und mit dem Passwort „VDE8417“ öffnen können. In der Normenbibliothek finden Sie die Beilage im Abschnitt „Zusatzmaterial“ am Ende der Norm.

Die beigelegte digitale Beilage enthält:

- eine xsd-Datei mit der XML-Schema-Definition (XSD) SML sowie
- eine xml-Datei SML_MeterData_Example.

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. DIN und DKE sind nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Das Original-Dokument enthält Bilder in Farbe, die in der Papierversion in einer Graustufen-Darstellung wiedergegeben werden. Elektronische Versionen dieses Dokuments enthalten die Bilder in der originalen Farbdarstellung.

1 Anwendungsbereich

Diese Norm ist anwendbar für ein Kommunikationsprotokoll für Anwendungen im Umfeld der Datenbeschaffung und Parametrierung von Geräten.

Zielsetzung bei der Ausarbeitung der Norm war der primäre Wunsch, eine möglichst einfache, auch zur Implementation auf leistungsschwachen Embedded Systems geeignete Struktur zu finden, die für die Datenbeschaffung über Weitverkehrsstrecken genutzt werden kann.

Vor diesem Hintergrund wurde die „Smart Message Language (SML)“ geschaffen.

In Bild 1 ist die Verortung im Protokoll-Stapel und der Bezug zu anderen Protokollen aus dem gleichen Anwendungsbereich dargestellt.

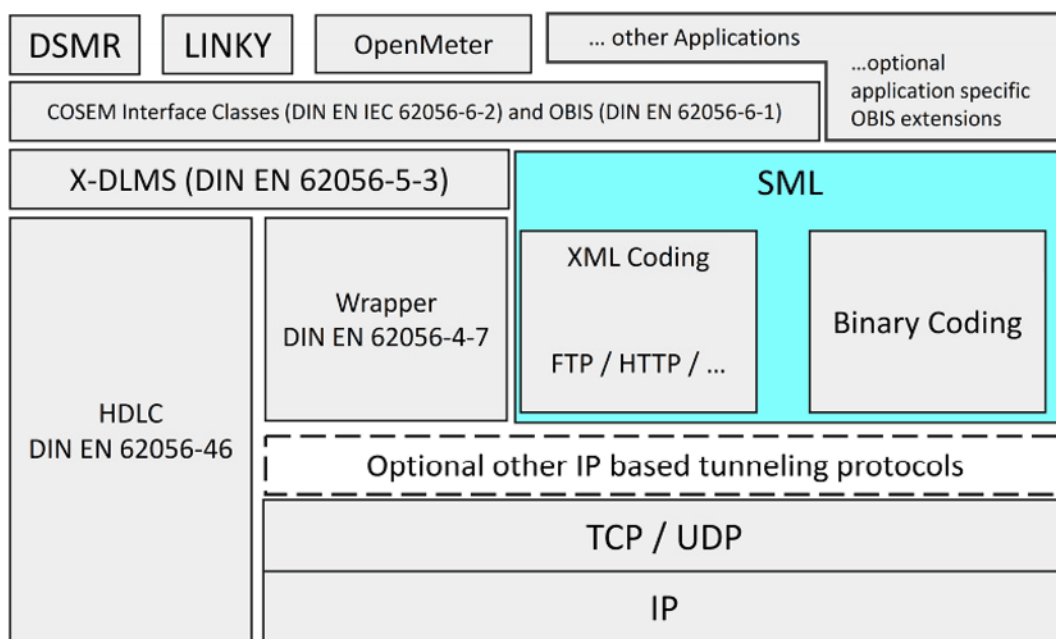


Bild 1 – Einordnung von SML in das Umfeld europäischer / internationaler Normen

2 Normative Verweisungen

Die folgenden Dokumente werden im Text in solcher Weise in Bezug genommen, dass einige Teile davon oder ihr gesamter Inhalt Anforderungen des vorliegenden Dokuments darstellen. Bei datierten Verweisungen gilt nur die in Bezug genommene Ausgabe. Bei undatierten Verweisungen gilt die letzte Ausgabe des in Bezug genommenen Dokuments (einschließlich aller Änderungen).

DIN EN 62056-46, *Messung der elektrischen Energie – Zählerstandsübertragung, Tarif- und Laststeuerung – Teil 46: Anwendung des HDLC-Protokolls in der Verbindungsschicht*

DIN EN 62056-6-1, *Datenkommunikation der elektrischen Energie – DLMS/COSEM – Teil 6-1: COSEM Object Identification System (OBIS)*

DIN EN IEC 62056-6-2, *Datenkommunikation der elektrischen Energie – DLMS/COSEM – Teil 6-2: COSEM Interface-Klassen*

DIN 43863-5, *Herstellerübergreifende Identifikationsnummer für Messeinrichtungen*

ISO/IEC 8859-15, *Informationstechnik – 8-Bit-Einzelbyte-codierte Schriftzeichensätze – Teil 15: Lateinisches Alphabet Nr. 9*

3 Begriffe und Abkürzungen

3.1 Begriffe

In diesem Dokument werden keine Begriffe aufgeführt.

DIN und DKE stellen terminologische Datenbanken für die Verwendung in der Normung unter den folgenden Adressen bereit:

- DIN-TERMinologieportal: verfügbar unter <https://www.din.de/go/din-term>
- DKE-IEV: verfügbar unter <http://www.dke.de/DKE-IEV>

3.2 Abkürzungen

Einheiten: Hinsichtlich physikalischer Messgrößen und Einheiten gelten die im SI getroffenen Vereinbarungen.

Relevante Abkürzungen: Den folgenden Abkürzungen können arabische Ziffern nachgestellt werden, um mehrfach auftretende Ausprägungen derselben Funktion/desselben Signals unterscheiden zu können.

+A	Wirkenergie, Kunde bezieht aus Netz
-A	Wirkenergie, Kunde liefert an Netz
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CMS	Cryptographic Message Syntax
CR	Carriage Return
DLMS	Device Language Message Specification
IANA	Internet Assigned Numbers Authority
ID	Identifikationsnummer
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LAN	Local Area Network
LSB	Least Significant Bit, de: niedrigstwertiges Bit
MDE	mobile Datenerfassungseinrichtung
MSB	Most Significant Bit, de: höchstwertiges Bit
OBIS	Objekt-Identifikations-System
OBIS-T	OBIS-Telemetrie
OSI	Open Systems Interconnection Reference Model
R1	Blindenergie Quadrant I
R2	Blindenergie Quadrant II
R3	Blindenergie Quadrant III
R4	Blindenergie Quadrant IV
RFC	Request for Comment
RS232	serielle Schnittstelle
S-I	Sekunden-Index
SML	Smart Message Language

TAG	Merkmal / Kennzeichnung / Auszeichnung bei der Kodierung von Datenelementen
TCP	Transmission Control Protocol
TL	Type-Length
TLZ	taktsynchroner Lastgangzähler
UDP	User Datagram Protocol
WAN	Wide Area Network
XML	Extensible Markup Language

4 Smart Message Language (SML)

4.1 Struktur

Die Struktur gliedert sich in folgende Elemente:

- Smart Message Language: Definiert eine Dateistruktur/Dokumentstruktur zur Aufnahme der zwischen den Endpunkten zu übertragenden Nutzlasten.
- SML Binary Encoding: Definiert eine gepackte binäre Kodierung der SML.
- SML XML Encoding: Definiert die Kodierung von SML in XML.
- SML-Transport-Protokoll: Benötigt für serielle Punkt-zu-Punkt Verbindungen.

SML-Nachrichten können, wie auch eine E-Mail, über zustandslose, gesicherte Kommunikationswege transportiert werden. Für das avisierte Einsatzszenario kann daher folgendes Modell zur Übersicht herangezogen werden (siehe Bild 2):

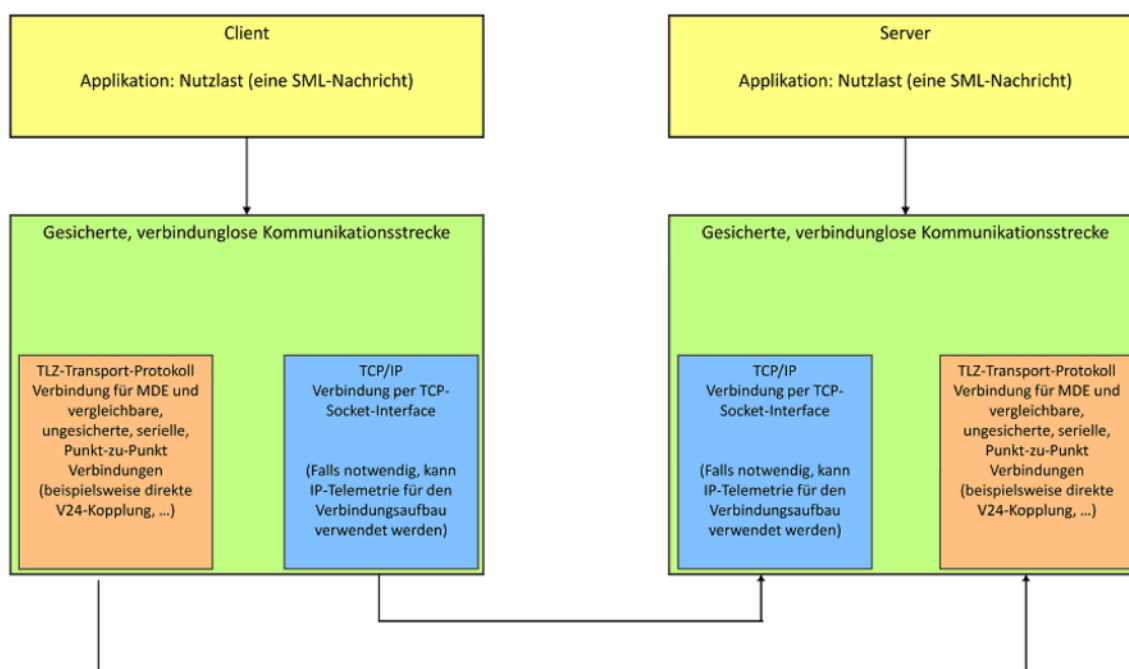


Bild 2 – SML-Nachrichten und Kommunikationswege

4.2 SML-Datei

Als SML-Datei wird eine Informationseinheit verstanden, die, vollkommen losgelöst von der jeweils eingesetzten konkreten Transporttechnik (Internet, Telefon, usw.), in sich abgeschlossen ist.

SML-Dateien können in diesem Sinne als abgeschlossene Informationseinheiten aufgefasst werden, die, genau wie eine E-Mail, in ein Protokoll eingebettet sind und übertragen werden (siehe Bild 3).

Durch den Ansatz der Verwendung von SML-Dateien wird das Konzept unabhängig von der Aufgabe, konkrete Protokolle zum Informationsaustausch definieren zu müssen. Stattdessen wird lediglich verlangt, in einem konkreten Einsatzfall ein bestimmtes Protokoll (z. B. HTTP, FTP, usw.) auszuwählen und dieses sachdienlich zu parametrieren.

Soweit SML-Dateien als Dateien auf Rechnersystemen verwendet werden, sind diese Dateien ohne Einsatz zusätzlicher Rahmen und unter Verwendung der mit 5.4 definierten Kodierung zu notieren, es sei denn, die konkrete Applikation trifft explizit eine anders lautende Vorgabe.

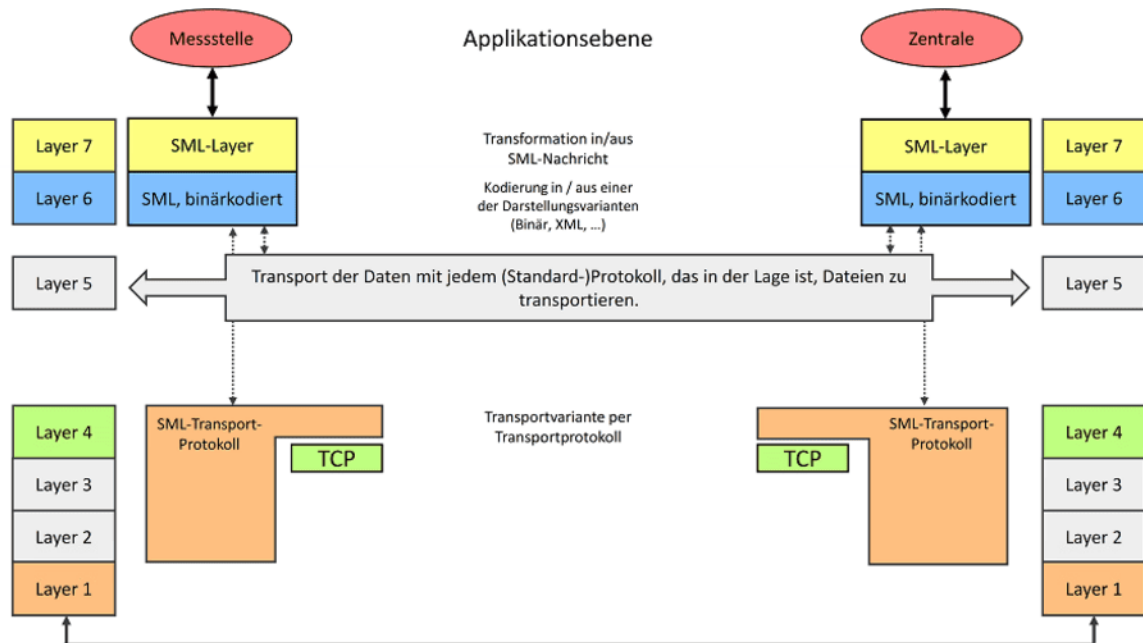


Bild 3 – SML-Kommunikationsmodell

4.3 Grundaufbau von SML-Nachrichten

Eine SML-Datei ist immer als Kette von SML-Nachrichten aufgebaut. SML-Dateien können zur Reduktion der Dateigröße segmentiert werden.

SML-Dateien können in den Varianten:

- SML-Auftragsdatei,
- SML-Antwortdatei oder
- SML-Kombidatei

auftreten.

SML-Auftragsdateien enthalten die Aufträge („Requests“). SML-Antwortdateien fassen die Antworten („Responses“) zu den Aufträgen zusammen.

Jede SML-Auftragsdatei beginnt mit genau einer und enthält genau eine SML_...Open.Reg-Nachricht. Sie endet mit genau einer und enthält genau eine SML_...Close.Reg-Nachricht. Diese Festlegung gilt ebenfalls für SML-Antwortdateien, denen keine SML-Auftragsdatei zugeordnet werden kann, wobei an Stelle der Variante Request die Variante Response zu verwenden ist.

SML-Antwortdateien, denen SML-Auftragsdateien zugeordnet werden können, beginnen mit genau einer SML_...Open.Res- oder einer SML_Attention.Res-Nachricht. Sie enden mit genau einer SML_...Close.Res- oder einer SML_Attention.Res-Nachricht.

Zum Einsatz von SML über Transportmedien mit geringer Performance¹, können SML-Nachrichten ohne den Rahmen einer SML-Datei und ausschließlich als „Response without Request“ versendet werden. Dieser Anwendungsfall ist explizit von der Applikation zu definieren.

Eine SML-Kombidatei enthält die SML-Nachrichten einer SML-Auftragsdatei zuzüglich der SML-Nachrichten der zugehörigen SML-Antwortdatei(en).

5 SML-Nachrichten

5.1 Allgemeines

Eine SML-Nachricht ist entweder eine ‚Request-Nachricht‘ oder eine ‚Response-Nachricht‘.

Eine SML-Nachricht umfasst Aufgabe und zugeordnete Attribute:

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

Die ‚transactionId‘ wird bei der Erzeugung von ‚Request-Nachrichten‘ in ein-eindeutiger Form durch den Auftraggeber gebildet. Jede ‚Response-Nachricht‘ spiegelt die zu deren ‚Request‘ gehörende Transaktionsnummer unverändert zurück, so dass eine ‚Response‘ stets dem zugehörigen ‚Request‘ zugeordnet werden kann. Die Eindeutigkeit ist daher jeweils innerhalb des Umfelds des Erzeugers der Transaktionsnummer zu gewährleisten; nicht gefordert wird eine weltweite Eindeutigkeit (vergleichbar der MAC-Adresse bei Ethernet). Durch die Angaben in der SML-Open-Nachricht kann die Eindeutigkeit auf den jeweiligen Erzeuger zugeordnet werden.

Wird im Sinne des ‚Push-Betriebs‘ eine SML-Datei erzeugt, zu deren SML-Response-Nachrichten prinzipbedingt keine SML-Request-Nachrichten existieren, erzeugt der Ersteller dieser SML-Response-Nachricht selbsttätig ein-eindeutige Transaktionsnummern.

Das Attribut ‚groupNo‘ erlaubt die Bildung von SML-Nachrichten-Gruppen. Dieser Mechanismus soll dazu verwendet werden, anzugeben, welche SML-Nachrichten in einer bestimmten Reihenfolge abzuarbeiten sind und welche nebenläufig (und zwar sowohl im Sinne von Simultanarbeit der ersten wie auch der zweiten Art) ausgeführt werden dürfen. In [Tabelle 1](#) ist die Verwendung exemplarisch dargestellt.

Generell erfolgt die Abarbeitung der Gruppen sequenziell und in der Reihenfolge, in der sie innerhalb einer SML-Datei auftreten.

Das Vermischen von SML-Nachrichten verschiedener Gruppen ist unzulässig.

Nachrichten innerhalb einer Gruppe können vom Empfänger wahlweise seriell oder nebenläufig abgearbeitet werden.

Die Prüfsumme (Element ‚crc16‘) ist als CRC16 nach DIN EN 62056-46 zu berechnen. Die Berechnung beginnt mit dem ersten Byte zu ‚SML_Message‘ und endet mit dem letzten Byte zu ‚messageBody‘. Damit sind die Bytes der Elemente ‚crc16‘ und ‚null‘ von der Prüfsummenberechnung ausgeschlossen.

¹ Im Sinne dieses Dokuments gelten als „Transportmedien mit geringer Performance“ ausschließlich folgende Varianten: Nahfunk- oder PLC-Strecken zwischen Sensoren und Konzentratoren.

Wird eine SML-Request-Nachricht empfangen, die dekodiert werden konnte und deren Prüfsumme fehlerhaft ist, so ist diese SML-Request-Nachricht mit einem ‚SML_Attention‘ (81 81 C7 C7 FE 0B, siehe [Tabelle 2](#)) und dem Feld ‚transactionId‘ (siehe [5.1](#)) zu beantworten. Wird in diesem Fall die Nachricht ‚SML_Close-Request‘ empfangen, so ist der Fehler zu ignorieren und in der Antwort ein korrektes ‚SML_Close-Request‘ zu liefern, dessen Feld ‚transactionId‘ (siehe [5.1](#)) gesetzt wird. Wird in diesem Fall die Nachricht ‚SML_Close-Request‘ nicht empfangen (stellt der Empfänger also ein von der Applikation zu definierendes Timeout fest), so ist diese SML-Request-Nachricht mit einem ‚SML_Attention‘ (81 81 C7 C7 FE 14, siehe [Tabelle 2](#)) und dem Feld ‚transactionId‘ (siehe [5.1](#)) zu beantworten.

Wird die Nachricht ‚SML_Open-Request‘ mit fehlerhafter Prüfsumme oder fehlerhafter Struktur ‚SML_PublicOpen.Reg‘ empfangen, so ist die ganze SML-Datei zu ignorieren.

Wird eine SML-Request-Datei empfangen, die dekodiert werden konnte und die nicht mit ‚SML_Open-Request‘ sondern einer anderen SML-Message beginnt, so ist die ganze SML-Datei zu ignorieren, d.h. der empfangene Inhalt wird verworfen und es wird keine Antwort generiert.

Wird eine SML-Nachricht empfangen, die nicht dekodiert werden kann, so ist wie folgt zu verfahren:

- Handelt es sich bei der SML-Nachricht um die erste Nachricht, so wird die ganze SML-Datei verworfen (gilt sowohl für SML-Auftragsdateien als auch für SML-Antwortdateien).
- Handelt es sich um eine der folgenden SML-Nachrichten, so wird als Antwort für die betroffene SML-Nachricht eine SML-Attention (81 81 C7 C7 FE 01, siehe [Tabelle 2](#)) gefolgt von einem SML-Close gesendet (gilt nur für SML-Auftragsdateien). Alle weiteren SML-Nachrichten der betroffenen SML-Datei werden ignoriert (sowohl bei SML-Auftragsdateien als auch bei SML-Antwortdateien).
- Die ‚transactionId‘ ist für die SML-Attention sowie für das sich anschließende SML-Close (siehe [5.1](#)) zu erzeugen.

SML-Nachrichten werden grundsätzlich in SML-Dateien zusammengefasst (siehe [4.3](#)).

Tabelle 1 – Beispiel zur Verwendung des Merkmals ‚Gruppen-Nummer‘

Reihenfolge der SML-Nachrichten	Transaktionsnummer	Gruppen-Nr.	SML-Nachricht	Kommentar
0	0	1	OPEN	Wird zuerst ausgeführt.
1	1	4	GET_ProfilPack	Erste Gruppe, wird als zweiter Block ausgeführt. Beispielsweise drei Lastgänge ablesen.
2	2	4	GET_ProfilPack	
3	3	4	GET_ProfilPack	
4	4	7	GET_ProfilPack	Zweite Gruppe, wird als dritte Aktion ausgeführt. Beispielsweise einen Lastgang, die Verrechnungsliste und das Logbuch ablesen.
5	5	7	GET_ProfilList	
6	6	7	GET_ProfilList	
7	7	8	SET_ProcParameter	Wird als vierte Aktion ausgeführt. Beispielsweise den Bezugszeitpunkt nachführen
8	8	8	CLOSE.Reg	Wird zuletzt bearbeitet.

Damit ist die Transaktionsnummer lediglich das Merkmal, um die Responses den zugehörigen Requests zuzuordnen. Sie hat keinerlei Einfluss auf die Reihenfolge der Ausführung der SML-Nachrichten beim Empfänger.

Das Attribut ‚abortOnError‘ legt fest, wie im Falle von Fehlern bei der Ausführung der SML-Nachricht verfahren werden soll, wobei das abschließende Close immer ausgeführt werden muss und nicht ausgeführte Requests immer eine Response mit der Fehlermeldung ‚nicht ausgeführt‘ als Rückmeldung erzeugen:

abortOnError	<=> 0x00	Ausführung fortsetzen,
abortOnError	<=> 0x01	Ausführung ab der nächsten Gruppe fortsetzen,
abortOnError	<=> 0x02	Ausführung ab der aktuellen Gruppe fortsetzen, danach keine weitere Gruppe mehr ausführen. Falls innerhalb derselben Gruppe ein ,0x02' zu finden ist, auf das ein ,0x01' folgt, ist die Ausführung sofort abzubrechen.
abortOnError	<=> 0xFF	Ausführung sofort abbrechen.

Ein SML Message Body besteht aus einer der folgenden Datenstrukturen:

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	
AttentionResponse	[0x0000FF01]	SML_Attention.Res	
}			

5.2 SML-Bezeichner

5.2.1 SML_PublicOpen.Req

Der SML_PublicOpen.Req muss immer zu Beginn einer SML-Auftragsdatei vorhanden sein. Er dient der Identifikation des Auftraggebers und der Authentifizierung per Benutzer/Passwort sowie der Zuordnung von SML-Antwortdatei(en) an die SML-Auftragsdatei.

Jeder Auftragnehmer muss auf den SML_PublicOpen.Req entweder mit einer SML_PublicOpen.Res oder einer SML_Attention.Res antworten. Die leere ‚Server-ID‘ ist als Broadcast-Adresse für zu beantwortende Nachrichten anzusehen. Verwendet eine Anwendung abweichend dazu Nachrichten mit speziellen Broadcast- oder Multicast-Adressen, sind diese explizit in der Anwendung zusammen mit dem Verhalten zu definieren.

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‘ wird, sofern der Wert angegeben ist, eine andere als die Default-Codepage vereinbart. Die ‚Codepage‘ legt fest, welcher Zeichensatz zur Interpretation von Zeichenketten zu verwenden ist. Fehlt der Wert, wird ISO/IEC 8859-15 verwendet.

Der Inhalt zur ‚Codepage‘ selbst ist immer als Zeichenkette in ISO/IEC 8859-15 zu notieren.

Der Parameter ‚serverId‘ erlaubt es, sofern der Wert angegeben ist, per ‚PublicOpen.Req‘ im Sinne einer Adresse gezielt eine SML-Datenquelle (ein konkretes Messgerät) oder ein Softwaremodul anzusprechen. In diesem Fall müssen alle folgenden SML-Nachrichten ebenfalls den Parameter ‚serverId‘ mit Inhalt angeben. Fehlt der Parameter ‚serverId‘ in einer der folgenden Nachrichten, so ist für die entsprechende SML-Nachricht mit einem ‚SML_Attention‘ (81 81 C7 C7 FE 0C, siehe [Tabelle 2](#)) zu antworten.

Fehlt der Parameter ‚serverId‘, wird die Anfrage als ‚Broadcast‘ gewertet. In diesem Fall muss bei allen folgenden SML-Nachrichten der Parameter ‚serverId‘ ebenfalls fehlen. SML-Nachrichten, die in dieser Situation dennoch den Parameter ‚serverId‘ angeben, sind zu ignorieren. In jedem Fall ist das abschließende ‚SML_Close‘ zu beantworten.

Der Parameter ‚clientId‘ wird bei der Erzeugung einer SML-Auftragsdatei vom Auftraggeber erzeugt und dient der eindeutigen Adressierung der SML-Antwortdatei des Clients.

Der Parameter ‚reqFileId‘ bezeichnet in systemweit eindeutiger Form ein konkretes SML-Auftragsdatei-/SML-Antwortdatei-Tupel. Er erlaubt es, SML-Antworten zu deren Aufträgen zuzuordnen. Die Eindeutigkeit ist jeweils innerhalb des Umfelds des Erzeugers der ‚reqFileId‘ sicherzustellen; nicht gefordert wird eine weltweite Eindeutigkeit (vergleichbar der MAC-Adresse bei Ethernet). Durch die anderen Angaben in der SML-Open-Nachricht kann die Eindeutigkeit auf den jeweiligen Erzeuger zugeordnet werden.

Die ‚reqFileId‘ liefert die eindeutige Kennzeichnung der SML-Datei, beispielsweise gebildet aus dem aktuellen Zeitstempel.

Fehlt der Parameter ‚smlVersion‘, wird die Version 1 als Standard angenommen.

5.2.2 SML_PublicOpen.Res

Der SML_PublicOpen.Res steht immer zu Beginn einer SML-Antwortdatei. Er dient der Identifikation der SML-Antwortdatei zu der zugehörigen SML-Auftragsdatei.

```

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,
}

```

Das Element ‚SML_Time‘ wird entweder als Sekunden-Index oder als Zeitstempel angegeben.

```

SML_Time ::= CHOICE
{
    secIndex              [0x01]      Unsigned32,
    timestamp             [0x02]      SML_TimeStamp,
    localTimeStamp        [0x03]      SML_TimeStampLocal
}

```

Handelt es sich um einen Zeitstempel, wird dieser immer in Sekunden ausgehend vom 01.01.1970, 00:00:00 (UNIX-Bezugszeitpunkt, bezogen auf UTC), gebildet.

```

SML_TimeStamp           ::= Unsigned32

```

Handelt es sich um eine lokale Zeitangabe, wird diese wie folgt notiert:

```

SML_TimeStampLocal      ::= SEQUENCE
{
    timestamp             SML_TimeStamp,
    localOffset           Integer16,
    seasonTimeOffset      Integer16
}

```

Das Element ‚localOffset‘ ist in Minuten anzugeben.

Das Element ‚seasonTimeOffset‘ ist in Minuten anzugeben.

Die lokale Zeit ergibt sich wie folgt:

$$\text{localTime} = \text{TimeStamp} + \text{localOffset} + \text{seasonTimeOffset}$$

Falls die Antwort von einem Gerät oder einem Softwaremodul geliefert wird, das selbst nicht über eine Zeitinformation verfügt, fehlt diese Angabe im ‚PublicOpen.Res‘.

Fehlt der Parameter ‚smlVersion‘, wird die Version 1 als Standard angenommen. Empfängt ein SML-Server eine ‚SML_PublicOpen.Req‘-Nachricht, in der der Parameter ‚smlVersion‘ eine SML-Version signalisiert, die der SML-Server nicht unterstützt, hat der SML-Server mit einer SML-Datei bestehend aus ‚SML_PublicOpen.Res‘ und ‚SML_Close.Res‘ zu antworten, wobei der Parameter ‚smlVersion‘ in der ‚SML_PublicOpen.Res‘-Nachricht auf jene SML-Version zu setzen ist, die der SML-Server unterstützt. Dieser Mechanismus erlaubt es einem SML-Client, sich bedarfsweise der vom SML-Server angebotenen SML-Version anzupassen.

Ist die SML-Datei eine SML-Response-Datei, zu der eine SML-Request-Datei gehört, wird das Element ‚reqFileId‘ der SML-Request-Datei zurückgegeben.

Ist die SML-Datei eine SML-Response-Datei, zu der keine SML-Request-Datei gehört, wird das Element ‚reqFileId‘, vergleichbar der Erzeugung dieses Attributs beim Erstellen einer SML-Request-Datei, auf einen systemweit eindeutigen Namen gesetzt.

Ist die SML-Datei eine SML-Response-Datei, zu der keine SML-Request-Datei gehört, darf das Element ‚clientId‘ weggelassen werden; in allen anderen Fällen ist der mit dem ‚PublicOpen.Req‘ angelieferte Wert dort einzutragen.

Das Feld ‚refTime‘ liefert den Referenzzeitpunkt zur Erstellung der SML-Antwortdatei.

5.2.3 SML_PublicClose.Req

Der SML_PublicClose.Req muss immer am Ende einer SML-Auftragsdatei vorhanden sein. Er beendet diese Datei.

Jeder Auftragnehmer muss auf den SML_PublicClose.Req mit einer SML_PublicClose.Res oder einer SML_Attention.Res antworten. Die leere ‚Server-ID‘ ist als Broadcast-Adresse für zu beantwortende Nachrichten anzusehen. Verwendet eine Anwendung abweichend dazu Nachrichten mit speziellen Broadcast- oder Multicast-Adressen, sind diese explizit in der Anwendung zusammen mit dem Verhalten zu definieren.

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

5.2.4 SML_PublicClose.Res

Der SML_PublicClose.Res steht immer am Ende einer SML-Antwortdatei. Er beendet diese Datei.

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

SML_Signature             ::= Octet String
```

5.2.5 SML_GetProfilePack.Req

In einer SML-Auftragsdatei kann eine oder können mehrere SML_GetProfilePack-Nachrichten vorhanden sein.

Jede SML_GetProfilePack.Req dient der Anfrage von einzelnen Messwerten oder Messwerte-Listen, die in gepackter Form übertragen werden.

Der Auftragnehmer muss auf jeden SML_GetProfilePack.Req mit genau einer SML_GetProfilePack.Res oder einer SML_Attention.Res antworten. Die leere ‚Server-ID‘ ist als Broadcast-Adresse für zu beantwortende Nachrichten anzusehen. Verwendet eine Anwendung abweichend dazu Nachrichten mit speziellen Broadcast- oder Multicast-Adressen, sind diese explizit in der Anwendung zusammen mit dem Verhalten zu definieren.