

DIN 43863-9

ICS 35.240.95; 91.140.50

Einsprüche bis 2019-02-26

Entwurf**Elektrizitätszähler –****Teil 9: Intelligentes Kommunikationsprotokoll für Stromzähler (SML)**

Electricity meter –

Part 9: Smart Message Language (SML)

Compteurs d'électricité –

Partie 9: Intelligent protocole pour les compteur de courant (SML)

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2018-10-26 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@vde.com möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE, Stresemannallee 15, 60596 Frankfurt am Main.

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 49 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Inhalt

	Seite
Vorwort.....	2
Abkürzungsverzeichnis	5
Normen	7
1 Bezug.....	8
2 Grundstruktur.....	8
3 Begriffe	9
3.1 SML-Datei.....	9
3.2 Byte.....	10
4 SML, Smart Message Language	10
4.1 Grundaufbau.....	10
5 SML-Nachrichten	11
5.1 SML-Bezeichner	14
5.1.1 SML_PublicOpen.Req	14
5.1.2 SML_PublicOpen.Res	15
5.1.3 SML_PublicClose.Req	16
5.1.4 SML_PublicClose.Res.....	16
5.1.5 SML_GetProfilePack.Req	16
5.1.6 SML_GetProfilePack.Res.....	18
5.1.7 SML_GetProfileList.Req	21
5.1.8 SML_GetProfileList.Res	21
5.1.9 SML_GetProcParameter.Req	22
5.1.10 SML_GetProcParameter.Res.....	23
5.1.11 SML_SetProcParameter.Req.....	25
5.1.12 SML_Attention.Res	25
5.1.13 Globale SML-Attentionnummer	26
5.1.14 SML_GetList.Req	27
5.1.15 SML_GetList.Res	28
5.2 Zugriff auf COSEM-Objekt per SML	28
5.3 Application-Container	33
5.3.1 Application-Container zur Inhaltsdatenverschlüsselung	34
5.3.2 Application-Container zur Ablösung von ‚SML_TupelEntry‘	37
6 SML binary encoding, direkt gepackte Kodierung.....	40
6.1 Type-Length-Field	40
6.2 Kodierung der Datentypen.....	41
6.2.1 Datentyp Octet String	41
6.2.2 Datentypen Integer8, Integer16, Integer32 und Integer64	41
6.2.3 Datentypen Unsigned8, Unsigned16, Unsigned32 und Unsigned64	42
6.2.4 Datentyp Boolean	43

	Seite
6.2.5 Datentyp List of ...	43
6.3 Kodierung besonderer Merkmale	43
6.3.1 Merkmal Ende einer SML-Nachricht	43
6.3.2 Merkmal SEQUENCE	44
6.3.3 Merkmal CHOICE	44
6.3.4 Merkmal OPTIONAL	44
7 XML - Kodierung	44
8 SML-Transport-Protokoll	44
8.1 Version 1	44
8.2 Version 2	46
8.2.1 Einleitung der Übertragung nach Version 2	46
8.2.2 Vereinbarung des zu verwendenden Timeouts	47
8.2.3 Prozess zum Aufbau der Übertragung	47
8.2.4 Prozess zum Ablauf einer Übertragung	47
8.2.5 Beispiel zum Ablauf des Übertragungsvorgangs nach Version 2	47
9 Benötigte Kennzahlen	48
Bilder	
Bild 1 – SML-Nachrichten und Kommunikationswege	8
Bild 2 – SML-Kommunikationsmodell	9
Bild 3 – Einordnung von SML in das Umfeld europäischer / internationaler Normen	10
Bild 4 – COSEM-Objekt-Attribut-Referenz(en) als Baum von Byte-Ketten im ‚SML_TreePath‘	29
Bild 5 – Attribut-Inhalt(e) als Baum von Byte-Ketten im ‚SML_Tree‘	30
Bild 6 – COSEM-Objekt-Methoden-Referenz als Baum von Byte-Ketten in einem ‚SML_Tree‘	33
Bild 7 – Anwendungshinweis zum Einsatz der Inhaltsdatenverschlüsselung	36
Tabellen	
Tabelle 1 – Liste referenzierter Normen und ergänzend benutzter Literatur	7
Tabelle 2 – Beispiel zur Verwendung des Merkmals ‚Gruppen-Nummer‘	12
Tabelle 3 – Liste globaler Fehlernummern	26
Tabelle 4 – Liste globaler Hinweisnummern	27
Tabelle 5 – Zuordnung von COSEM- und SML-Datentypen	31
Tabelle 6 – Application-Container zur Inhaltsdatenverschlüsselung	34
Tabelle 7 – Application-Container zur Ablösung von ‚SML_TupelEntry‘	37
Tabelle 8 – Bitkodierung im Type-Length-Field für das erste Byte einer TL-Field-Angabe	40
Tabelle 9 – Bitkodierung im Type-Length-Field für das zweite und folgende TL-Field-Bytes	41
Tabelle 10 – Bitkodierung im Type-Length-Field für einen Octet String	41
Tabelle 11 – Escape-Merkmale zum SML-Transport-Protokoll	45
Tabelle 12 – Kennzahlen-Liste zu den ‚Application Container‘-Datenstrukturen	49

Vorwort

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

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe des Dokuments.

Der Zusammenhang der zitierten Dokumente mit den entsprechenden deutschen Dokumenten ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

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.

Abkürzungsverzeichnis

Einheiten:

- (1) Hinsichtlich physikalischer Messgrößen und Einheiten gelten die im SI (siehe DIN 1301, Teil 1) getroffenen Vereinbarungen.

Relevante Abkürzungen:

- (2) Den nachfolgenden 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,
DIN	<=> Deutsches Institut für Normung e.V.,
DKE	<=> Deutsche Elektrotechnische Kommission im DIN,
DLMS	<=> Device Language Message Specification,
(E) DIN	<=> Entwurf einer Norm des DIN,
EN	<=> Europäische Norm,
IANA	<=> Internet Assigned Numbers Authority,
ID	<=> Identifikationsnummer,
IEC	<=> International Electrotechnical Commission,
IEEE	<=> Institute of Electrical and Electronics Engineers,
IP	<=> Internet Protocol,
ISO	<=> Internationale Organisation für Normung,
LAN	<=> Local Area Network,
LSB	<=> Least Significant Bit, niederwertigstes Bit,
MDE	<=> Mobile Datenerfassungseinrichtung,
MSB	<=> Most Significant Bit, höchstwertigstes Bit,
OBIS	<=> Objekt-Identifikations-System,
OBIS-T	<=> OBIS Telemetry,
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,
UDP	<=> User Datagram Protocol,
WAN	<=> Wide Area Network,
XML	<=> Extensible Markup Language,
ZVEI	<=> Zentralverband Elektrotechnik- und Elektronikindustrie.

Normen

Tabelle 1 – Liste referenzierter Normen und ergänzend benutzter Literatur

DIN 1301, Teil 1	10.02	Einheiten, Teil 1: Einheitennamen, Einheitenzeichen
DIN 43863-4	11.16	Zählerdatenkommunikation – IP-Telemetrie
DIN 43863-5	04.12	Geräte-Einzelidentifikation, Herstellerübergreifende Identifikationsnummer für Messeinrichtungen
DIN EN 62056-21	01.03	Elektrizitätszähler, Zählerstandsübertragung, Teil 21: Datenübertragung für festen und mobilen Anschluss (3rd edition of IEC 61107, vormals IEC 1107)
DIN EN 62056-61	01.03	Messung der elektrischen Energie – Zählerstandsübertragung, Teil 61: OBIS Objekt Identification System
IEC 62056-62 DIN EN 62056-62	2002 01/03	Messung der elektrischen Energie - Zählerstandsübertragung, Tarif- und Laststeuerung - Teil 62: Interface-Klassen (IEC 62056-62:2002)
DIN EN 62056-46	01/03	Messung der elektrischen Energie - Zählerstandsübertragung, Tarif- und Laststeuerung - Teil 46: Anwendung des HDLC-Protokolls in der Verbindungsschicht (IEC 62056-46:2002)
CCITT-CRC16	--.--	Standard der CCITT zur Prüfsummenberechnung
ISO 8859-15	03.99	Informationstechnik - 8-Bit-Einzelbyte-codierte Schriftzeichensätze - Teil 15: Lateinisches Alphabet Nr. 9
RFC 5652	06.13	Cryptographic Message Syntax (CMS) (http://www.rfc-editor.org)
IANA-TLS	05.14	Transport Layer Security (TLS) Parameters (http://www.iana.org/assignments/tls-parameters/tls-parameters.txt)

1 Bezug

- (3) Nachfolgende Spezifikation legt ein Kommunikationsprotokoll für Anwendungen im Umfeld der Datenbeschaffung und Parametrierung von Geräten fest.
- (4) Zielsetzung bei der Ausarbeitung der Spezifikation 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.
- (5) Vor diesem Hintergrund wurde die „Smart Message Language“, SML, geschaffen.

2 Grundstruktur

- (6) Die Grundstruktur gliedert sich in die 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.
- (7) SML-Nachrichten, siehe Kapitel 4, können, wie letztlich auch eine E-Mail, über zustandslose, gesicherte Kommunikationswege transportiert werden. Für das avisierte Einsatzszenario kann daher folgendes Modell zur Übersicht herangezogen werden:

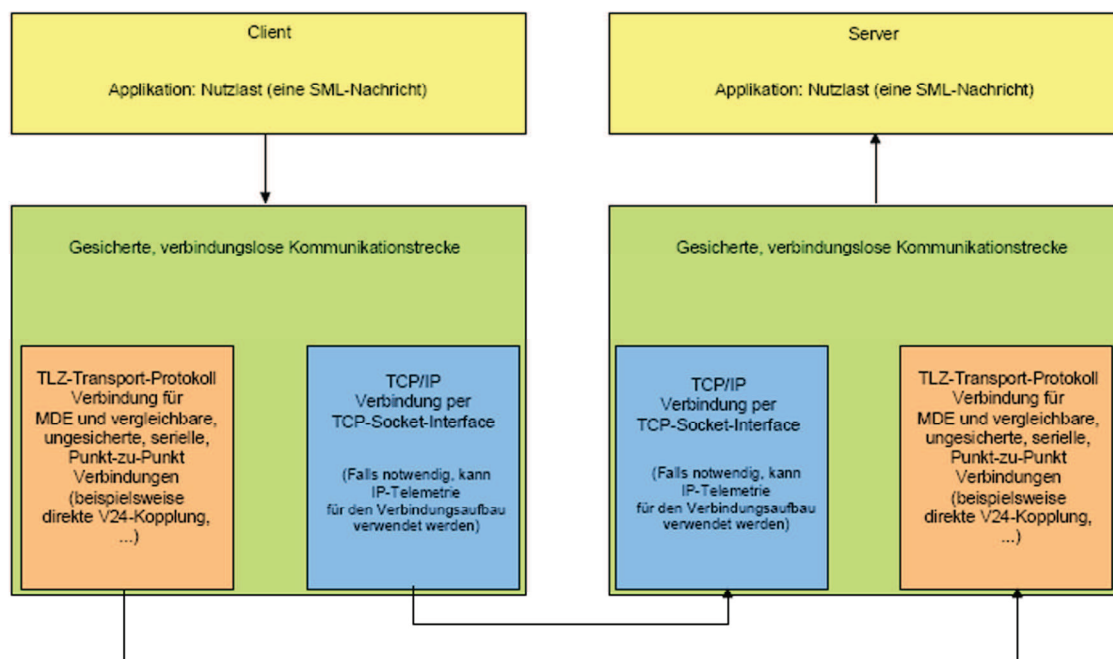


Bild 1 – SML-Nachrichten und Kommunikationswege

3 Begriffe

- (8) Nachstehend werden einige / wichtige der in diesem Dokument verwendeten Begriffe erläutert / definiert:

3.1 SML-Datei

- (9) Als SML-Datei soll eine Informationseinheit verstanden werden, die, vollkommen losgelöst von der jeweils eingesetzten konkreten Transporttechnik (Internet, Telefon, ...), in sich abgeschlossen ist.
- (10) 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 2).
- (11) 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 (beispielsweise HTTP, FTP, ...) auszuwählen und dieses sachdienlich zu parametrieren.
- (12) Soweit SML-Dateien als Dateien auf Rechnersystemen verwendet werden, sind diese Dateien ohne Einsatz zusätzlicher Rahmen und unter Verwendung der mit 5.3 definierten Kodierung zu notieren, es sei denn, die konkrete Applikation trifft explizit eine anders lautende Vorgabe.

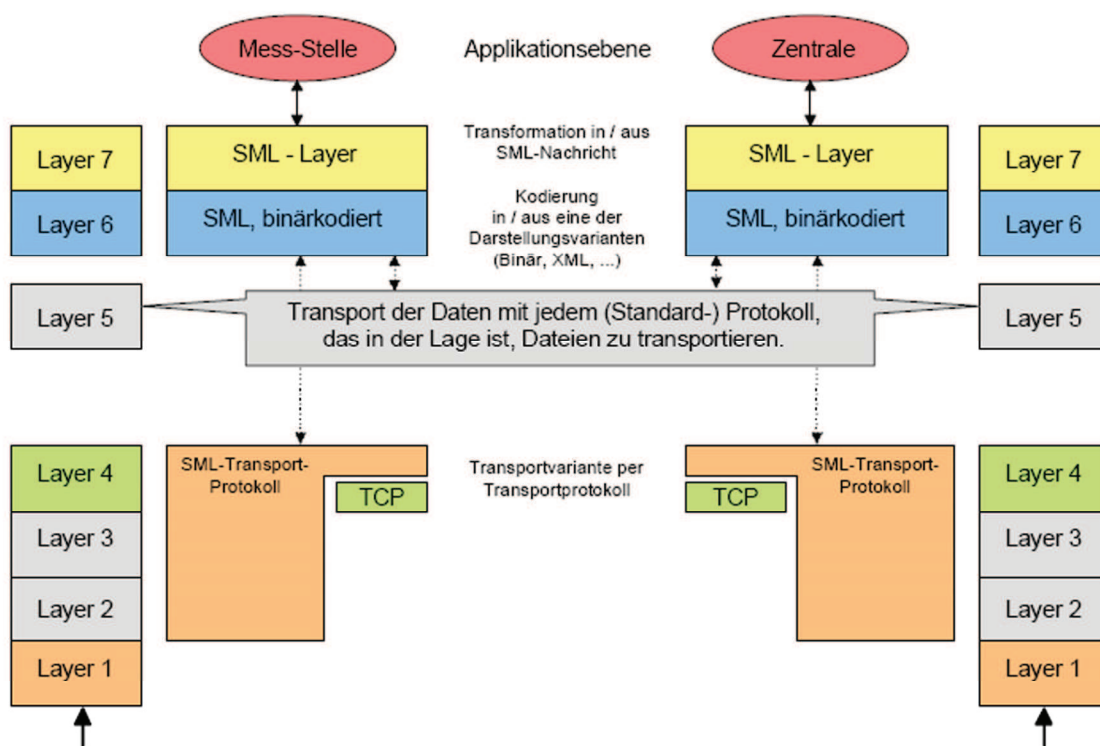


Bild 2 – SML-Kommunikationsmodell

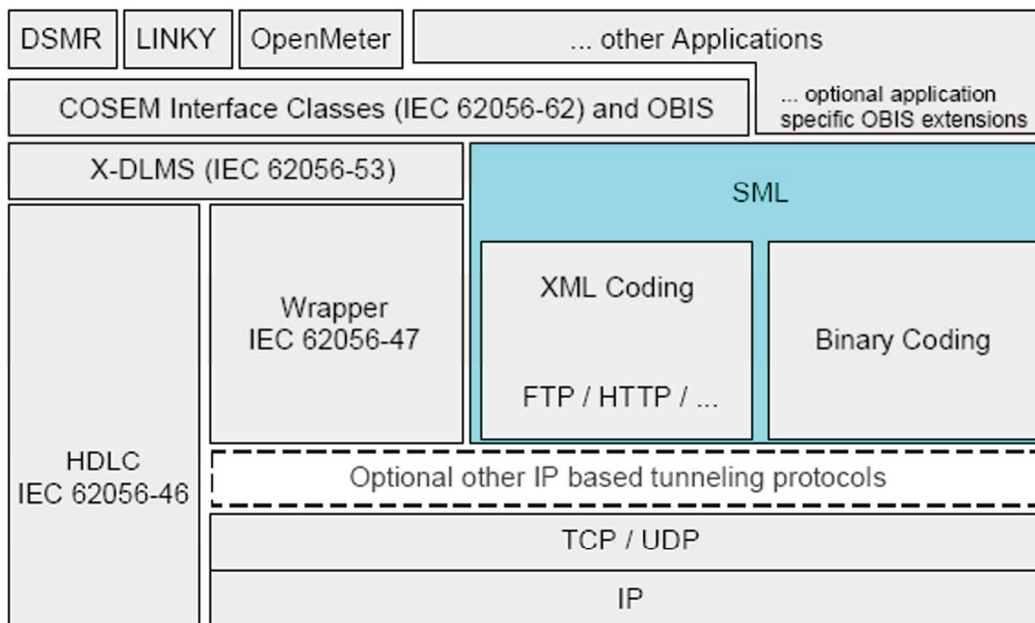


Bild 3 – Einordnung von SML in das Umfeld europäischer / internationaler Normen

3.2 Byte

- (13) In diesem Dokument wird die Einheit / der Begriff ‚Byte‘ synonym für ein Oktett und damit einer Abfolge von 8 Bit benutzt.

4 SML, Smart Message Language

4.1 Grundaufbau

- (14) Eine SML-Datei ist immer als Kette von SML-Nachrichten aufgebaut.
- (15) SML-Dateien können zur Reduktion der Dateigröße segmentiert werden.
- (16) SML-Dateien können in den Varianten...
 ... SML-Auftragsdatei,
 ... SML-Antwortdatei oder
 ... SML-Kombidatei
 auftreten.
- (17) SML-Auftragsdateien enthalten die Aufträge („Requests“). SML-Antwortdateien fassen die Antworten („Responses“) zu den Aufträgen zusammen.
- (18) Jede SML-Auftragsdatei beginnt mit genau einer und enthält genau eine SML_...Open.Res-Nachricht. Sie endet mit genau einer und enthält genau eine SML_...Close.Res-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.
- (19) 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.

- (20) 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.
- (21) Eine SML-Kombidatei enthält die SML-Nachrichten einer SML-Auftragsdatei zuzüglich der SML-Nachrichten der zugehörigen SML-Antwortdatei(en).

5 SML-Nachrichten

- (22) Eine SML-Nachricht ist entweder eine ‚Request-Nachricht‘ oder eine ‚Response-Nachricht‘.
- (23) Eine SML-Nachricht umfaßt Aufgabe und zugeordnete Attribute.

(A) SML_Message ::= SEQUENCE

```

{
    transactionId      Octet String,
    groupNo            Unsigned8,
    abortOnError       Unsigned8,
    messageBody        SML_MessageBody,
    crc16              Unsigned16,
    endOfSmlMsg        EndOfSmlMsg
}

```

- (24) Die ‚transactionId‘ wird bei der Erzeugung von ‚Request-Nachrichten‘ in ein-eindeutiger² 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.
- (25) 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.
- (26) 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.
- (27) Generell erfolgt die Abarbeitung der Gruppen sequentiell und in der Reihenfolge, in der sie innerhalb einer SML-Datei auftreten.
- (28) Das Vermischen von SML-Nachrichten verschiedener Gruppen ist unzulässig.
- (29) Nachrichten innerhalb einer Gruppe können vom Empfänger wahlweise seriell oder nebenläufig abgearbeitet werden.
- (30) 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 dieser Spezifikation gelten als „Transportmedien mit geringer Performance“ ausschließlich folgende Varianten: Nahfunk- oder PLC-Strecken zwischen Sensoren und Konzentratoren.

² Die Eindeutigkeit ist 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.

- (31) 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 Tab. 3) und dem Feld ‚transactionId‘ gemäß Absatz (25) 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‘ gemäß Absatz (25) 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 Tab. 3) und dem Feld ‚transactionId‘ gemäß Absatz (25) zu beantworten.
- (32) Wird die Nachricht ‚SML_Open-Request‘ mit fehlerhafter Prüfsumme oder fehlerhafter Struktur ‚SML_PublicOpen.Req‘ empfangen, so ist die ganze SML-Datei zu ignorieren³.
- (33) 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.³
- (34) 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 nachfolgenden SML-Nachrichten, so wird als Antwort für die betroffene SML-Nachricht ein SML-Attention (81 81 C7 C7 FE 01, siehe Tab. 3) 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 gemäß Absatz (25) zu erzeugen.
- (35) SML-Nachrichten werden grundsätzlich in SML-Dateien zusammengefasst.

Tabelle 2 – 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.Req	Wird zuletzt bearbeitet.

- (36) 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.

³ Ignorieren: Der empfangene Inhalt wird verworfen. Es wird keine Antwort generiert.

- (37) 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:

(B)	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 der selben
			Gruppe ein ‚0x02‘ zu finden
			ist, auf das ein ‚0x01‘ folgt, ist
			die Ausführung sofort
			abzubrechen.
	abortOnError	<=> 0xFF	Ausführung sofort abbrechen.

(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
	AttentionResponse	[0x0000FF01]		SML_Attention.Res
	}			

5.1 SML-Bezeichner

5.1.1 SML_PublicOpen.Reg

- (38) Der SML_PublicOpen.Reg 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.
- (39) Jeder Auftragnehmer muss auf den SML_PublicOpen.Reg 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.
- (D) SML_PublicOpen.Reg ::= 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, |
| } | |
- (40) 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 8859-15‘ verwendet.

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

- (41) Der Parameter ‚serverId‘ erlaubt es, sofern der Wert angegeben ist, per ‚PublicOpen.Reg‘ im Sinne einer Adresse gezielt eine SML-Datenquelle (ein konkretes Messgerät) oder ein Softwaremodul anzusprechen. In diesem Fall müssen alle nachfolgenden SML-Nachrichten ebenfalls den Parameter ‚serverId‘ mit Inhalt angeben. Fehlt der Parameter ‚serverId‘ in einer der nachfolgenden Nachrichten, so ist für die entsprechende SML-Nachricht mit einem ‚SML_Attention‘ (81 81 C7 C7 FE 0C, siehe Tabelle 3) zu antworten.
- (42) Fehlt der Parameter ‚serverId‘, wird die Anfrage als ‚Broadcast‘ gewertet. In diesem Fall muß bei allen nachfolgenden 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.
- (43) Der Parameter ‚clientId‘ wird bei der Erzeugung einer SML-Auftragsdatei vom Auftraggeber erzeugt und dient der eindeutigen Adressierung der SML-Antwortdatei des Clients.
- (44) Der Parameter ‚reqFileId‘ bezeichnet in systemweit⁴ eindeutiger Form ein konkretes SML-Auftragsdatei- / SML-Antwortdatei-Tupel. Er erlaubt es, SML-Antworten zu deren Aufträgen zuzuordnen.
- (45) Die ‚reqFileId‘ liefert die eindeutige Kennzeichnung der SML-Datei, beispielsweise gebildet aus dem aktuellen Zeitstempel.

⁴ Die Eindeutigkeit ist jeweils innerhalb des Umfelds des Erzeugers der ‚reqFileId‘ zu gewährleisten; 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.

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

5.1.2 SML_PublicOpen.Res

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

(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,
 }

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

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

(49) 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.

(G) SML_Timestamp ::= Unsigned32

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

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

(51) Das Element ‚localOffset‘ ist in Minuten anzugeben.

(52) Das Element ‚seasonTimeOffset‘ ist in Minuten anzugeben.

(53) Die lokale Zeit ergibt sich wie folgt:
 localTime = Timestamp + localOffset + seasonTimeOffset

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

(55) Fehlt der Parameter ‚smlVersion‘, wird die Version 1 als Standard angenommen. Empfängt ein SML-Server eine ‚SML_PublicOpen.Res‘-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