Bundesamt
für Sicherheit in der
Informationstechnik

Technische Richtlinie BSI TR-03109-1

Anlage II: COSEM/HTTP Webservices

Version 1.0, Datum 18.03.2012

Bundesamt für Sicherheit in der Informationstechnik

Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-100

E-Mail: SmartMeter@bsi.bund.de
Internet: https://www.bsi.bund.de

Smart Meter Gateway
Teil 1: COSEM/HTTP Webservices

Version 1.01
Stand: 06.03.2013 / Anlage zur TR-03109 1.0
Verantwortlich: DKE AK 461.0.143

Inhalt

Tabellen

Bilder

1. Dokumentenhistorie

Datum	Version	Änderung	Autor
3.8.2012		Erster Entwurf für Sitzung in Bonn am 6.8.2012	
17.8.2012		Kommentare eingearbeitet (u.a. Typos, CMS ext.Dokument, W3C KeyInfo, UseCases und Kommunikationsszenarien getrennt, Eigenes Kapitel für Festlegungen), Bewertung Client/Server-Betriebsarten	MSt,MWa,RPl,MCa,JWA
25.8.2012		Zwei Parallele TLS Verbindungen, Gateway ist immer TLS-Client, TLS-Compression und CMS-Kompression	AK143
31.10.2012		Cleaned up für TR03109	MSt
14.11.2012		Added COSEM over HTTP and cleaned up after BSI Workshop	AK143, MSt
17.11.2012		Kapitel 7 um Detail-Festlegungen zum HTTP-Access (Header etc.) erweitert . Asynchrone Response (x-ContactURI, x-CID). Collections→Containers	MSt
19.11.2012		Redaktionssitzung: Kapitel „HTTP-COSEM“, XSD und COSEM-XML Abbildung und als Spezifikation zur Referenzierung durch die TR03109 herausgezogen. Kapitel „Kommunikationsszenarien“, „UseCases“ und	MCa,RPl,HBa,MSt

		„TLS“ separiert.	
22.11.2012	0.80	Redaktionskreis: TR03109 Bezug in Anhang (Kap 9). Keine Referenzen auf TR03109 im Hauptteil.	MCa,RPl,MSt
23.11.2012	0.81	Gerätekennzeichnung Prefix wie LMN: de09-	MSt
27.11.2012	0.82	Webkonferenz AK 143: Keine xml_base Elemente, Selective-Access mit COSEM-Datentypen,	AK143
28.11.2012	0.83	Beispiel in 9.7 korrigiert	MWA
29.11.2012	0.84	Änderungen nach Telco AK143:Idempotenz/Safeness für Services, XSDs ausgearbeitet,Class-LN Notation festgelegt,Range/Blocktranssfer beschrieben	AK143,MSt
30.11.2012	0.85	Redaktionskreis, Editorielle Korrekturen	MSt,RPl,HBa
11.12.2012	0.86	AK 143 Sitzung, Frankfurt. Domain-Name-Prinzip für Gerätebezeichner (Postfix) /smgw und /gwa Element in den TR03109 Anhang verschoben Content-Encoding/Accept-Encoding Header für CMS	AK 143
17.12.2012	0.87	Content-Type und Content-Encoding überarbeitet.	MCa,MSt
8.2.2013	1.00	Editorielle Korrekturen nach Kommentierung	MSt
6.3.2013	1.01	Referenziert PP 1.2 und TR-03109v1.0	MSt

2. Referenzierte Dokumente

Folgende Schriftstücke werden durch dieses Dokument adressiert:

Ref.	Identifikation	Version	Dokument
[COSEM]	IEC 62056-62 DIN EN 62056-62	Entwurf 10.2011	Companion Object Specification for Energy Metering Messung der elektrischen Energie - Zählerstandsübertragung, Tarif- und Laststeuerung - Teil 62: Interface-Klassen
[CMS-ECKA]	TR03109 Anlage I	1.0	CMS-Datenformat für die Inhaltsdatenverschlüsselung und -signatur
[HTTP]	RFC2616	HTTP/1.1	Hypertext Transfer Protocol Version 1.1
[OBIS]	IEC 62056-61 DIN EN 62056-61	Entwurf 10.2011	Object Identification System Messung der elektrischen Energie – Zählerstandsübertragung, Teil 61: OBIS Objekt Identification System
[RFC-URI]	RFC 3986		Kodierung einer URI

Ref.	Identifikation	Version	Dokument
[TR]	TR 03109	1.0	Technische Richtlinie 03109,
[PP]	CC-PP-0073	1.2	BSI Schutzprofil für ein Smart Meter Gateway…
[TLS]	RFC 5246	TLS1.2	
[UUID]	RFC4122		A Universally Unique IDentifier (UUID) URN Namespace
[XMLBASE]	W3C xmlbase		http://www.w3.org/TR/xmlbase/
[DNSNAMES]	RFC1035	1987	Domain Implementation and Specification
[XSD-COD]	urn:k461-dke-de:cod-1	0.2	Entwurf des XSD Schemas für COSEM Datentypen nach DLMS Contribution 050, basierend auf asn.1
[XSD-COR]	urn:k461-dke-de:cor-1	0.2	Entwurf des XSD Schemas für RESTful COSEM Webservices des DKE AK461.0.143

Tabelle 1: Liste referenzierter Dokumente

3. Abkürzungen und Begriffe

Abkürzungen und Begriffe werden wie folgt benutzt:

APDU – Application Protocol Data Unit (Inhaltsdatenstruktur einer Nachricht)

Asset – Im Kontext des Schutzprofiles ein schützenswertes Gut (z.B. eine Datenstruktur oder ein Speicher (-inhalt))

Body /Content-Body – Ein HTTP Request und Response teilt sich in einen Header (Kopf) und einen Body (Rumpf). Der Body enthält den zu transportierenden Inhalt der Nachricht.

COSEM - Companion Specification for Energy Metering (Internationale Spezifikation zur beschreibt von Datenstrukturen/Klassen für das Messwesen)

CRUD – Create, Read, Update, Delete Methoden (Verben) zum Zugriff auf Resourcen

DNS – Domain Name System (Dienst zur Abbildung von Netzwerkadressen auf Domain Names und umgekehrt)

EMT – Externer Marktteilnehmer

FQDN – Fully Qualified Domain Name (www.example.com)

HES – Head End System (Terminiert die Kommunikationsverbindungen auf den Transport oder TLS-Layer). Betrieben von Kommunikationsdienstleistern (und Externen Marktteilnehmern).

HMAC – Hashed Message Authentication Code (eine mittels Geheimnis gebildete Prüfsumme)

HTTP – Hypertext Transfer Protocol

Objekt – Instanz einer Klasse. Ein COSEM-Objekt enthält COSEM-Attribute und COSEM-Methoden.

Ressource – Bezeichnet im HTTP/REST Kontext ein über den URI-Baum adressierbares Element. Dies kann ein Container oder im COSEM Kontext ein Phyiscal_Device, Logical_Device,Object, Attribut oder eine Methode sein.

Request – Die Anforderungsnachricht vom Client zum Server. Wird vom Server mit einer Response beantwortet.

Response – Die Antwortnachricht vom Server zum Client. Wird beim Client verarbeitet.

REST(ful) – Representational State Transfer (Stil eine API und einen Datenzugriff über Resourcen mit einheitlichem CRUD-Model abzubilden)

RFC – Request for comments. IETF Spezifikation oder Standard.

SMGW – Smart Meter Gateway

SMGW_ADM – Smart Meter Gateway Administrator

TR - Technische Richtlinie

Verb – Im Kontext von HTTP eine Zugriffsmethode: GET, PUT, POST, DELETE

4. Einleitung

4.1. Scope

Dieses Dokument beschreibt:

- Eine Zugriffs- und Transportschicht für COSEM-Objekte über HTTP (Webservices alternativ zu IEC62056-53)
- Abbildung von Identifizierungselementen aus COSEM (Logical_Name, Logical_DeviceID, Attribute_ID) zur Verwendung in HTTP
- Anforderungen an den darunterliegenden Transportlayer
- Eine allgemeine Abbildung von COSEM-Datentypen und Datenstrukturen auf XML mittels eines XSD Schemas.

Ein Anhang dieses Dokumentes beschreibt die Einschränkenden Vorgaben zur Anwendung dieser Spezifikation zur Verwendung mit der BSI [TR] und dem Schutzprofil für ein Smart Meter Gateway [PP].

4.2. Dokumentenstruktur

Die Struktur des Dokumentes orientiert sich an den OSI-Protokoll-Layern:

Layer	Name	Standard	Kapitel
7.2	Identifikation, Datenmodell	OBIS, TR-03109	
7.1	Klassen, Datenstrukturen	COSEM	
6	Repräsentation/Datentypen	XML+XSD(ASN.1)	Kapitel 8
5S	Content Encoding, Encryption, Signature, Compression	CMS	Optional CMS with ECKA-EG
5.2	Request/Response, Content Description	HTTP-Header+Body	Kapitel 7
5.1	Verb	HTTP-Verb+Noun	Kapitel 7
4S	Transport Security	TLS	TR-03109
4	Transport Layer	z.B. TCP	
3	Network Layer	z.B. IP	
2	Media Access/Data Link Layer		
1	Media Dependent/Physical Layer		

Tabelle 2: Einordnung in OSI Layer

Der Grüne Bereich wird in dieser Spezifikation beschrieben.

4.3. Anwendungsbereich

Die vorliegende Spezifikation beschreibt die Abbildung der in den Standards [COSEM] (COSEM-Interface-Classes) und [OBIS] bereitgestellten Definitionen über HTTP mittels eines RESTful API Stiles.

Die Spezifikation ist Teil der in Arbeit befindlichen Beschreibung zum Smart-Meter-Gateway (siehe Festlegungen des BSI sowie der Verbände). Es ist beabsichtigt, diese Ergänzungen künftig in die international geltenden Standards einzubringen.

Die Spezifikation ist im Protokollstack alternativ zu IEC62056-53 angeordnet.

4.4. Übersicht über wesentliche Anforderungen an die Spezifikation

1. HTTP/1.1 wird als Webservice-Protokoll verwendet.
2. Abbildung von COSEM-Interface-Klassen über HTTP. Zusätzlich kann über die Webservice-Schnittstelle Administration (z.B. File-Download) und Mehrwertdienste auch ohne COSEM-Klassen-Abbildung genutzt werden (dies ist nicht mehr Teil dieser Spezifikation).
3. Abbildung des COSEM-IC Zugriffes über RESTful Verben und URI-Adressierung
4. Der Inhalt des HTTP-Body ist abhängig vom Content-Type/Accepted und soll auch Inhaltsdatengesicherter Transport ermöglichen. Der Client kann dem Server mitteilen, welche Inhaltsdatentypen er akzeptiert.
5. Ermöglichen der für den Betrieb von intelligenten Messystemen notwendigen UseCases und den daraus abgeleiteten Kommunikationsszenarien: MANAGEMENT, ADMIN-SERVICE, INFO-REPORT.
6. Einschränkung der Zugriffsrechte auf Server-Ressourcen abhängig von der Authentifizierung des Clients.
7. Auftrennung der Client/Server und Server/Client Funktion auf HTTP-Ebene zwischen Gateway und Administrator in zwei parallele Verbindungen.

- Die HTTP-Session wird zusammen mit der darunterliegenden (sicheren) Transportverbindung auf- und abgebaut.
- Die HTTP- Session muss nicht nach jeder Transaktion (Request/Response) beendet werden, sondern kann bestehen bleiben (Persistent Connection).

5. Datenmodell und API-Struktur

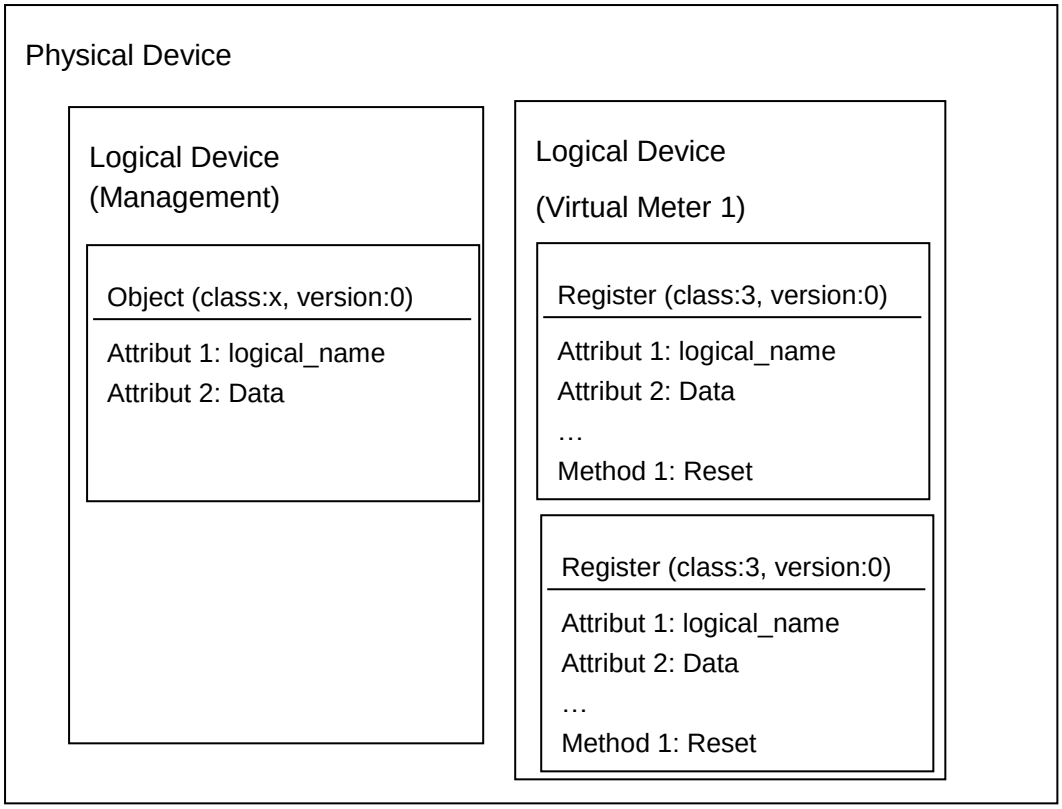
5.1. COSEM Physical Device, Logical Device, Object, Attributes, Methods,

Pro Physical COSEM Device MUSS ein Management Logical Device vorhanden sein.

Pro Management Logical Device MUSS ein Logical-Device-Name Objekt vorhanden sein.

Die Verwendung des Management Logical Device muss noch spezifiziert werden.

Abbildung 1: Struktur des COSEM Physical Device



5.2. URI Resource-Tree eines Physical COSEM Devices

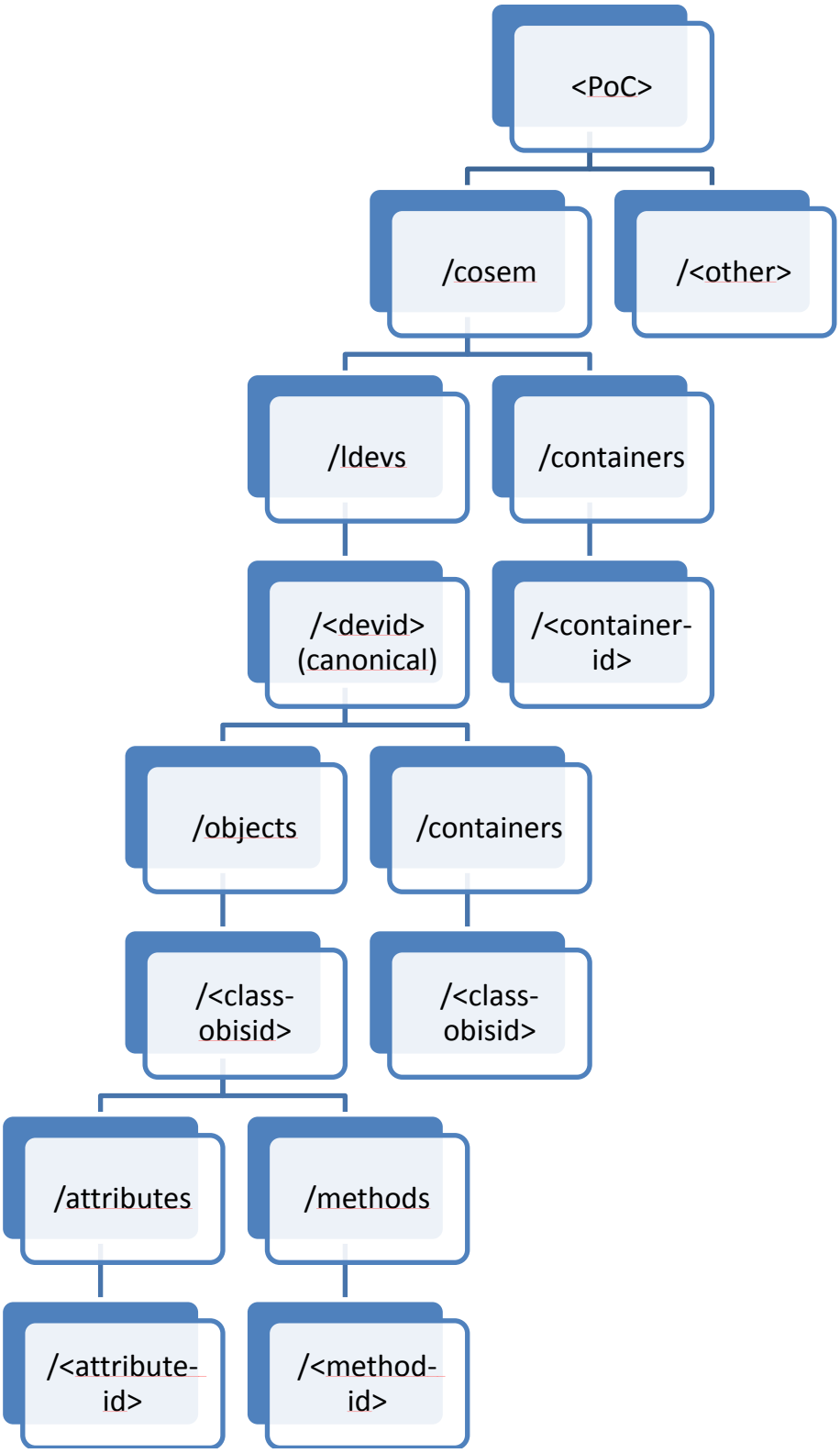


Abbildung 2: URI Baum

Der URI Baum spiegelt die Struktur des Datenmodelles wieder. Im RESTful Stil wird der Zugriffsdienst (Read/Write/Action) auf Bestandteile des Datenmodelles (Ressourcen) über http-Verben durchgeführt.

6. Abbildung von Identifikatoren und Bezeichnern

6.1. Kanonischer Gerätebezeichner

Zur Verwendung verschiedener Gerätenummerierungsschemen in dieser Spezifikation wird ein kanonischer Gerätebezeichner definiert. Dieser Gerätebezeichner enthält nur die Zeichen 0-9,-,a-z und hat eine Länge von 1-63 Zeichen. Damit ist er geeignet als Hostname Bestandteil des DNS Systems zu sein.

Für den kanonischen Gerätebezeichner gelten die Bildungsregeln eines DNS-Namens [DNSNAMES] mit den oben genannten Einschränkungen. Der Gerätebezeichner MUSS im Sichtbarkeitsbereich des Gateways und seiner Kommunikationspartner eindeutig sein.

Die Identifikation des Nummernschemas MUSS aus dem Domain-Namen erkennbar sein.

Wird kein FQDN verwendet, muss zwischen Client und Server eine Vereinbarung der gemeinsamen übergeordneten Domain getroffen vorliegen.

Hinweis: Obwohl eine DNS-Notation verwendet wird, besteht keine Notwendigkeit eine Namensauflösung der Gerätebezeichner auf Transport-Layer Adressen über DNS durchzuführen.

Nummerierungsschema	Postfix	Kanonisierung
DIN43863-5:2012-04	.sm	Kleinbuchstaben, keine Leerzeichen
MAC (EUI-48/EUI-64)	.eui	Kleinbuchstaben, keine Trennzeichen

Tabelle 3: Postfix für Nummerierungsschema

6.2. Hostname

Der HTTP-Hostname soll für das Gateway als Kanonischer Gerätebezeichner gebildet werden.

6.3. Logical_DeviceID

Die Logical_DeviceID für Geräte soll als Kanonischer Gerätebezeichner gebildet werden. Es ist das Percent-Encoding aus [RFC-URI] Kap. 2.1 anzuwenden, wenn nicht darstellbare oder reservierte Zeichen in der Logical_DeviceID enthalten sind.

Hinweis: Die Logical_DeviceID wird in [COSEM] als Datentyp octet-string oder visible-string spezifiziert. Durch das XSD Schema [XSD-COD] wird eine XML-Repräsentation als hexBinary angegeben.

Beispiel:
Logical_DeviceID: 0x01 MFC1234
URI: %01MFC1234
XML: 014d464331323334

6.4. COSEM-Objekt-Bezeichner in der URI

COSEM Objekte innerhalb eines Logical_Devices werden eindeutig durch das Tupel COSEM-Class-ID und Logical-Name beschrieben. Die Version wird zur Adressierung in der URI nicht verwendet. Das Tupel { Class-ID, Logical_Name } MUSS als Konkatenation von Class-ID (als Dezimalzahl 0-65535), Bindestrich, und dem Logical_Name in hexBinary Repräsentation notiert werden. Im Identifikator sind nur die Zeichen 0-9, Bindestrich und a-f erlaubt. Die Länge beträgt damit 14 bis 18 Zeichen.

Diese Spezifikation verwendet als Logical_Name OBIS-IDs nach [OBIS] mit den dort beschriebenen länderspezifischen Erweiterungen.

Das Byte für OBIS Value-Group A steht am Anfang des Strings. Fehlt der Wert für Value-Group F enthält die Repräsentation ff und wird nicht verkürzt.

Beispiel:
/objects/3-0100010800ff/attributes/3

6.5. COSEM Attribut-Index

Jedes COSEM-Objekt enthält mindestens ein Attribut (logical_name).

Index Dezimal	Attribut
-128..-1	Herstellerspezifische Klassenerweiterung
0	Nicht vorhanden (z.T. Sonderfall: „Alle Attribute>0“)
1	Logical_Name (OBIS-ID)
2..127	Attribute die in der COSEM-Klasse definiert sind

Tabelle 4: Attribut Indizes

Die Repräsentation der Attribut-Indizes MUSS als Dezimalzahl im Bereich -128 bis 127 angegeben werden

6.6. COSEM Methoden-Index

Ein COSEM-Objekt kann Klassen-Spezifische Methoden enthalten. Jede Methode hat einen pro Klasse eindeutigen Index. Der Index der ersten Methode eines Objektes ist 1.

Die Repräsentation der Methoden-Indizes MUSS als Dezimalzahl im Bereich -128 bis 127 angegeben werden

6.7. Container Bezeichner in der URI

Bezeichner für aggregierter Objekte verwenden die Bildungsregel nach Kap. 6.4, d.h. ein Tupel aus Class-ID und Logical-Name.

7. COSEM Zugriff über HTTP

7.1. HTTP-Request/Response

Diese Spezifikation basiert auf der Protokollspezifikation von HTTP/1.1 [HTTP] und nimmt einige Einschränkungen und wenige Erweiterungen vor.

Der grundsätzliche Ablauf einer HTTP-Client/Server Kommunikation wird kurz dargestellt:

Ein Gerät (Client-Rolle) sendet eine Anfrage (Request) an seinen Kommunikationspartner (Server-Rolle). Der Server sendet darauf immer eine Antwort (Response) an seinen Kommunikationspartner (Client).

Eine Response ohne Request ist nicht möglich.

Beispiel für einen HTTP-Request:

```
GET /path/resource HTTP/1.1<CRLF>
Header1: Tokens<CRLF>
Headern: Tokens<CRLF>
<CRLF>
Optionaler Inhaltsteil ("Body")
```

Beispiel für einen innerhalb der gleichen Verbindung erzeugten Response:

```
HTTP/1.1 Statuscode Statustext<CRLF>
Header1: Tokens<CRLF>
Headern: Tokens<CRLF>
<CRLF>
Optionaler Inhaltsteil ("Body")
```

Die Länge der Request-URI soll nicht größer als 4000 Zeichen sein. .

Die Verwendung von HTTP/1.1 ermöglicht die Aufrechterhaltung der Verbindung nach einer Response, so dass der nächste Request innerhalb der Verbindung gesendet wird. Durch einen Connection-Header kann das Schließen der Verbindung nach dem Response angekündigt werden.

Pipelineing von Requests ist nicht erlaubt, d.h. bevor ein neuer Request innerhalb einer HTTP-Session gesendet wird muss eine Response vorliegen.

Anmerkung: Ziel dieses APIs ist Client und Server auf HTTP-Ebene möglichst Zustandslos betreiben zu können. D.h. Zwischen Request und Response wird keine Information beim Client gespeichert, die für die Zuordnung der Antwort notwendig ist. Zustandslosigkeit führt zu einer besseren Skalierbarkeit der Infrastruktur bei einer großen Zahl von Client/Server Verbindungen.

7.1.1. HTTP Response Timeout

Der Response-Timeout MUSS konfigurierbar sein. Der Defaultwert hängt von der Verbindungsart ab (z.B. 20s).

Ist nach Ablauf der konfigurierbaren Timeoutzeit keine Response eingetroffen, SOLL die HTTP-Session, und die Transportverbindung von beiden Seiten getrennt werden.

Kann der Server nicht innerhalb der Timeoutzeit eine synchrone Antwort liefern, wird abhängig von den Vorbedingungen später eine asynchrone Antwort an den Client gesendet oder eine Fehlermeldung an den Client übermittelt..

7.1.2. Nutzung von mehr als einer Verbindung zwischen Kommunikationspartnern

In dem hier verwendeten strikten Client/Server Konzept hat ein Kommunikations-Endpunkt immer nur eine HTTP-Rolle (Client oder Server). Um einem Gerät mit Server-Funktion zu ermöglichen eine asynchrone Nachricht an das Gerät mit Client-Funktion zu senden muss ein weiterer Kommunikationskanal aufgebaut werden in dem die Endpunkt-Rolle Client/Server gegenüber dem vorher beschrieben vertauscht sind.

7.2. Zuordnung von Zugriffs-Diensten und HTTP-Verben

Diese Spezifikation beschreibt nur den COSEM-Zugriff über Long-Names (LN-Addressing). Damit ergeben sich die Basis-Dienste Get, Set, Action. Diese müssen verpflichtend bereitgestellt werden.

Zur Abbildung von dynamischen Datenmodellen werden zusätzlich Dienste zum Anlegen und Löschen von Ressourcen definiert.

Zugriffs-Dienst	HTTP-Verb	Request-URI contains	Request-Body	Response-Body	Idempotenz /Safe
Get	GET	Container-, Objekt-, Attribut-Deskriptor	Leer	Container-, Objekt-, or Attribut-Werte	Idempotent, Safe
Set	PUT	Container-, Objekt-Attribut-Deskriptor	Container-, Objekt-, Attribute-Werte	Leer	Idempotent, Nicht Safe
Action	POST	Objekt-, Methoden-Deskriptor	Methoden Aufruf-Werte	Methoden Antwort Werte	Nicht Idempotent, Nicht Safe
Create	PUT	Container-, Objekt-Deskriptor	Container-, Object-, Attribut-Werte	Leer	Idempotent, Nicht Safe
Delete	DELETE	Container-, Objekt-Deskriptor	Leer	Leer	Idempotent, Nicht Safe

Notify (Event)	POST	Methoden-Deskriptor	Event-Parameter	Leer	N/A

Tabelle 5: Zugriffs-Dienste

Diese Tabelle listet sowohl die wesentlichen COSEM-Zugriffsdienste als auch die für Webservice notwendigen Zugriffsdienste.

Die Abbildung von Read-With-List und Write-With-List wird über Container-Zugriffe abgebildet, d.h. vor dem Zugriff wurde eine Container-Ressource (z.B. durch die Geräte Firmware) angelegt. Durch den Container ist es möglich alle Objekte in einer atomaren Operation zu lesen oder zu schreiben.

GET, PUT, POST, DELETE operieren genau auf der adressierten Ressource (und evtl. untergeordneten Ressourcen). Bei GET, POST und DELETE muss die Ressource existieren. Bei PUT wird die Ressource angelegt, falls sie nicht existiert.

	GET	PUT	POST	DELETE
Safe	Ja	Nein	Nein	Nein
Idempotent	Ja	Ja	Nein	Ja

Tabelle 6: Idempotenz und Safeness von HTTP-Verben

7.3. HTTP-Header

Die Anwendung von HTTP im Rahmen dieser Spezifikation erfordert nicht die Implementierung aller möglichen HTTP-Header die nach [HTTP] möglich sind. Für die Richtungen Request und Response werden im Folgenden sinnvolle Einschränkungen beschrieben:

Unbekannte Header-Felder MÜSSEN ignoriert werden. Der Vergleich der Header-Bezeichner MUSS Case-insensitiv durchgeführt werden (Gross/Kleinschreibung ist irrelevant)

Die Länge einer Header-Field Zeile (incl. CR,LF) SOLL NICHT mehr als 255 Zeichen betragen.

Alle weiteren Header aus [HTTP] KÖNNEN vorhanden sein.

7.3.1. REQUEST-HEADER

Header Name	M/K/B	RFC2616	Beschreibung
Content-Type	B	14.17	Sofern ein Request-Body nach dieser Spezifikation vorhanden ist, ist dieser Header verpflichtend. Gibt den Typ den Request-Bodies an.
Content-Encoding	B	14.11	Sofern ein Request-Body vorhanden ist, ist dieser Header verpflichtend, falls ein Content-Encoding angewandt wurde (wie z.B. Kompression)
Content-	M	14.13	Gibt die Länge des Request-Bodies in Bytes an.

Length			
x-CID	K		Correlation-ID zwischen Anfrage und Antwort. Dies ist besonders für asynchrone Antworten notwendig.
Host	M	14.23	Identifiziert den Host an den die Anfrage gerichtet ist. Dies entspricht der host-Identifikation aus der HTTP-URI.
Accept	K	14.1	Teilt dem Server mit, welche Content-Types im Body einer Response erlaubt sind.
Accept-Encoding	K	14.3	Teilt dem Server mit, welche Content-Encodings im Body einer Response erlaubt sind.
Range	K	14.35	Teilt dem Server mit welcher Teil einer Ressource übertragen werden soll.
x-ContactURI	K		URI, an die bei Asynchronen Responses der Notify-Request zugestellt werden kann.

Tabelle 7: Response Header

M:Muss, B:Bedingt, K:Kann

Requests in denen Muss Header Felder fehlen, müssen mit HTTP-Error 400 Bad Request abgelehnt werden.

Mit x- beginnende Header Felder sind nicht in [HTTP] vorhanden und werden in dieser Spezifikation definiert.

7.3.1.1. Content-Type Header

Gibt den Typ des Inhalts der Ressource im HTTP-Body an.

Der Wert ist der zu einem Inhalt festgelegte MIME-Type ggf. mit Parametern.

Content-Type	Inhalt	Beschreibung
application/octet-stream	Binärdaten	Z.B. File-Image
application/vnd.<orgid>+xml{;parameters}	XML File mit COSEM Daten	COSEM Daten in XML

Tabelle 8: Content-Types

Content-Type Parameter	Wert	Beschreibung
encap	cms-tr03109	CMS mit ECKA-EG nach BSI [TR]

Tabelle 9: Content-Type Parameter

Beispiel:

Content-Type: application/vnd.de-dke-k461-cosem +xml;encap=cms-tr03109

7.3.1.2. Content-Encoding Header

Der Content-Encoding Header beschreibt die Codierung des http-Bodies.

Der Content-Encoding Header muss angegeben werden, wenn der http-Body codiert wird.

Encoding	Inhalt
deflate	Kompression vor der Encapsulation

Tabelle 10: Content-Encodings für Content-Encoding/Accept-Encoding

7.3.1.3. Content-Length Header

Gibt die Länge des folgenden HTTP-Bodies in Bytes an. Dieses Feld MUSS im Request vorhanden sein.

Beispiel:

```
Content-Length: 100
```

7.3.1.4. Host Header

Enthält den Host-Namen des durch den Request adressierten Servers. Das Feld MUSS im Request vorhanden sein.

7.3.1.5. Accept Header

Mit diesem HTTP-Header–Feld teilt der Client dem Server mit, welche Content-Typen er im HTTP-Body der Response akzeptiert.

Sofern eine zusätzliche Spezifikation beschreibt welche Inhaltsdatentypen vom Client akzeptiert werden müssen, KANN der Accept Header entfallen.

7.3.1.6. Accept-Encoding Header

Mit diesem HTTP-Header–Feld teilt der Client dem Server mit, welche Content-Encodings er im HTTP-Body der Response akzeptiert.

Sofern eine zusätzliche Spezifikation beschreibt welche Content-Encodings vom Client akzeptiert werden müssen, KANN der Accept-Encoding Header entfallen.

7.3.1.7. x-CID Header

Über die Correlation-ID aus diesem Header-Feld wird der Bezug zwischen Anfrage und Antwort in asynchronen Antworten hergestellt.

Die Correlation-ID wird als uuid nach [UUID] angegeben. Diese weltweit mit sehr hoher Wahrscheinlichkeit eindeutige Nummer darf keine Rückschlüsse auf den Absender zulassen. Die Bildungsvorschrift „Version 4“ mit Zufallszahlengenerator ist anzuwenden.

Das Feld KANN angegeben werden, um eine Asynchrone Antwort zu ermöglichen.

Beispiel:

```
x-CID: uuid:f50701ab-5a39-4fa6-853f-58aef4297228
```

7.3.1.8. Range Header

Über das Range-Header-Field kann der Client dem Server mitteilen, dass nur ein Teil einer Ressource gelesen oder beschrieben werden soll (nur für GET/PUT Requests).

Damit kann der Transfer eines großen CMS-Body (Inhaltsdatengesicherter Container) nach Abbruch der Transportverbindung fortgeführt werden.

Das Feld KANN angegeben werden um einen Teil einer Ressource zu lesen oder zu schreiben.

Beispiel:

```
Range: bytes=1024-2047
```

7.3.1.9. x-ContactURI Header

Über den x-ContactURI-Header teilt der Client dem Server im Request mit, an welche URI (PoC) auf Client-Seite die asynchrone Antwort zugestellt werden KANN. Die Asynchrone Antwort wird per Notify (POST-Request) oder Set (PUT) zugestellt. Alternativ kann die Asynchrone Antwort an eine vorher hinterlegte Adresse+PoC zugestellt werden. Ist keine Adresse hinterlegt und keine x-ContactURI im Header übergeben, ist eine asynchrone Antwort nicht möglich. Kann die Anfrage nicht rechtzeitig beantwortet werden, wird die Anfrage mit einer Fehlermeldung `408 Request Timeout` beendet.

Beispiel:

```
x-ContactURI: https://<hostaddr>/hes/cosem/ldevs/<mgmtdev>/objects/
<eventobject>/methods/1
```

7.3.1.10. HTTP Kompression

Sowohl Server als auch Client MUSS unkomprimierten HTTP Body unterstützen.

Hinweis: Falls ein Verschlüsselter Inhaltstyp im http-Body verwendet wird ist eine Kompression vor der Verschlüsselung sinnvoll.

7.3.2. RESPONSE-HEADER

Header Name	M/K/B	RFC2616	Beschreibung

Content-Type	B	14.17	Sofern ein ResponseBody nach dieser Spezifikation vorhanden ist, ist dieser Header verpflichtend. Gibt den Typ den Response-Bodies an.
Content-Encoding	B	14.11	Sofern ein ResponseBody vorhanden ist, ist dieser Header verpflichtend, falls ein nicht Content-Encoding angewandt wurde. (z.B. Kompression)
Content-Length	B	14.13	Gibt die Länge des Response-Bodies in Bytes an.
Content-Range	K	14.16	Teilt dem Client mit, welcher Teil (Byte-Range) bei einer mit Range fragmentierten Anfrage in der Response geliefert wird.

Tabelle 11: Response-Header

M: MUSS, B:Bedingt, K:KANN

7.3.2.1. Content-Type Header

Gibt den Typ des Inhalts im HTTP-Body an. Siehe 7.3.1.1

7.3.2.2. Content-Encoding Header

Gibt das Encoding des Inhalts im HTTP-Body an. Siehe 7.3.1.2

7.3.2.3. Content-Length Header

Gibt die Länge des folgenden HTTP-Bodies in Bytes an. Dieses Feld MUSS angegeben werden, wenn ein Content-Body vorhanden ist. Fehlt der Content-Length Header wird ein leerer Body angenommen. Um „keinen Content“ zu signalisieren wird der HTTP-Status-Code `204 No Content` übermittelt.

7.3.2.4. Content-Range Header

In der Antwort auf einen Request mit Range-Header-Field liefert der Server den HTTP-Status-Code `206 Partial Content` und gibt im Content-Range-Field an welchen Teil (in Bytes) einer Ressource der Content-Body enthält.
Der Content-Range Header KANN in der Response vorhanden sein.

Beispiel:
```
Content-Range: bytes 1024-2047/2048
```

7.3.2.5. Retry-after Header

- Der Server teilt in der Antwort nach einem (Request-Timeout-) Fehler mit, wann der Client frühestens einen neuen Request schicken soll.
- Nur die Variante „Delta-Seconds“ soll verwendet werden

Beispiel:

```
Retry-After: 60
```

7.4. HTTP Status-Codes

Status Code	Statustext	Beschreibung
1xx	Intermediate Response	Der Client wartet nach Empfang dieses Statuscodes auf eine weitere Response vom Server.
200	OK	Request wurde erfolgreich ausgeführt.
201	Created	Resource wurde erfolgreich angelegt
202	Accepted	Asynchrone Antwort folgt durch Notification an die im Request übergebene x-ContactURI
204	No content	Signalisiert dass der Response-Body keinen Inhalt enthält. Der Content-Length Header fehlt. Z.B. nach Delete einer Ressource.
206	Partial content	Zeigt an, das der Response-Body nur einen Teil einer Response enthält (Content-Range Header ist vorhanden)
400	Bad request	Der Server kann den Request nicht ausführen, da z.B. verpflichtende Header Felder fehlen
403	Forbidden	Falscher Hostname
404	Not found	Pfad/Ressource nicht gefunden
405	Method not allowed	Für die ausgewählte Ressource ist Methode im Request nicht erlaubt (z.B. POST auf /cosem/ldevs)
406	Not acceptable	Die vom Client im Accept-Header übermittelten Content-Typen passen nicht zur Ressource. Der Server unterstützt nur application/pkcs7-mime;*
408	Request timed out	Die Antwort konnte nicht rechtzeitig bereitgestellt werden. Anfrage später nochmal wiedeholen.
411	Length required	Die Angabe von Content-Length ist verpflichtend.
413	Request entity too long	Body Content-Length ist zu gross und kann vom Server nicht verarbeitet werden.
414	Request URI too long	Die Länge der URI im Request ist zu lang und kann vom Server nicht verarbeitet werden.
416	Requested Range Not satisfiable	Der im Range-Header übergebene Bereich passt nicht zur Ressource.
500	Internal Server Error	Ein COSEM-Anwendungsfehler ist aufgetreten. Der Response-Body kann detaillierte Informationen (Event-Code) zum Fehler enthalten. COSEM-Fehler werden ins System Log geschrieben wodurch sie über Asynchrone – Meldung an den Client gemeldet werden können.
501	Not implemented	Die Methode (z.B. HEAD, OPTIONS) ist nicht implementiert.
506	HTTP Version not supported	NHTTP/1.1 muss vom Server unterstützt werden.

Tabelle 12: HTTP-Statuscodes

Der Statustext ist nicht normativ und soll nicht für Vergleiche herangezogen werden. Nur der Statuscode wird verglichen.

Die Statuscodes 2xx MUSS als positive Rückmeldung verstanden werden. Nach 202 und 206 ist die Kommunikations-Anfrage noch nicht abgeschlossen. Die Positiv-Rückmeldung kann noch nicht an die Anwendung kommuniziert werden bis nicht `200 ok` empfangen wurde.

Die Statuscodes 3xx (Redirection) SOLLEN NICHT verwendet werden und führen zu einem Abbruch der Transaktion.

Die Statuscodes 4xx sind Fehler im Zugriff des Clients und MUSS zum Abbruch der Transaktion mit Fehler führen. Der Fehler wird an die Anwendung gemeldet. Einige Fehler ermöglichen einen Retry nach einer Wartezeit.

Die Statuscodes 5xx sind Fehler des Servers und MÜSSEN an die Anwendungsschicht weitergeleitet werden.

Nach dem Empfang eines Fehlerstatuscodes kann der Client

1. Weitere Informationen zum Fehler mit einem weiteren (Polling-) Request dem Fehlerspeicher des Servers auslesen.
2. Den Fehler als Event über eine http-Verbindung an den Requester senden, beispielsweise wenn der Fehler eine konfigurierte Meldeschwelle überschreitet.
3. Hinweise zum Fehler können als COSEM-Datenstruktur im Response-Body enthalten sein.

7.5. Synchrone/Asynchrone Antwort

Ist der Server nicht in der Lage kurzfristig (Siehe 7.1.1) eine Antwort auf eine Anfrage zurückzuliefern, KANN er den Vorgang mit einer HTTP-Fehlermeldung abbrechen oder durch den Statuscode `202 Accepted` mitteilen, dass die Antwort später bereitsteht.

Um eine Antwort Asynchron zurückzusenden, MUSS ein x-CID Header (Correlation ID) im Request vorhanden sein.

Um eine Antwort Asynchron zurückzusenden MUSS der Client zuvor dem Server eine ContactURI bekannt gemacht haben. Dies kann durch Konfiguration oder durch das x-ContactURI Header-Feld geschehen,

Über den x-ContactURI-Header teilt der Client dem Server im Request mit, welchen Point-of-Contact auf Client-Seite die asynchrone Antwort behandelt. Die **asynchrone Antwort wird per Notify (POST-Request)** zugestellt.

Die Zuordung von Request- und Asynchroner Response (Notification) beim Client wird über die Correlation-ID (x-CID) vollzogen.

Beispiel der Kommunikation zwischen einem Gateway und einem Head-End-System:

Im ersten Kommunikationskanal (Gateway ist Server, Head-End-System ist Client)

Beispiel Head-End-System sendet Request an Gateway:

```
POST <GW-PoC>/cosem/…/methods/1 HTTP/1.1
```

```
x-ContactURI: https://<hostaddr>/<HES-PoC>/cosem/.../methods/2
x-CID: uuid: f50701ab-5a39-4fa6-853f-58aef4297228
(More Headers)

(Content-Body enthält Methoden Input Parameter)
```

Gateway sendet Response an Head-End-System:

```
HTTP/1.1 202 Accepted
Content-Length: 0
```

Nachdem der Methoden-Aufruf beendet ist und die asynchrone Antwort bereitsteht, übermittelt das Gateway die Antwort an das Head-End-System im aufgebauten Kommunikations-Kanal (Gateway ist:Client, Head-End-System ist Server):

Gateway → Head-End-System:

```
POST /<HES-PoC> /cosem/…/methods/2 HTTP/1.1
x-CID: uuid: f50701ab-5a39-4fa6-853f-58aef4297228
(Content Body enthält Methoden Output-Parameter/Resultat)
```

Head-End-System → Gateway:

```
HTTP/1.1 200 OK
Content-length: 0
```

7.6. Zugriffsrechte

Der Zugriff auf einzelne Ressourcen des Servers wird über die Authentifizierung der unterliegenden Transport-Verbindung (z.B. TLS-Zertifikat) eingeschränkt, dies ist nicht Bestandteil dieser Spezifikation und wird an anderer Stelle festgelegt. Der Server erlaubt nur Zugriffe mit den HTTP-Methoden GET, PUT, POST, DELETE.

Die exponierten Ressourcen die über HTTP-URI erreichbar sind, sind pro Kommunikationsverbindung unterschiedlich. Welche Ressourcen der Server den Client anbietet, entscheidet sich durch die Authentifizierung des Clients.

Ein Zugriff auf eine nicht exponiert Ressource die in der aktiven Verbindung nicht erreichbar ist MUSS mit einem Fehlercode `404 Not found` beantwortet werden.

Eine weitergehende Client-Authentifzierung innerhalb von HTTP durch www-authenticate-Header ist nach dieser Spezifikation nicht vorgesehen.

7.7. Repräsentation von COSEM Literalen in der HTTP-Zugriffsschicht

Literale von COSEM Datentypen werden folgendermaßen in der URI codiert.

COSEM Datatype	Abbildung auf Repräsentation	Beispiel
null-data(0)	<Leer>	null-data:
boolean(3)	Integer8	boolean:1
bit-string(4)	Octetstring	bit-string:1234

double-long(5)	Integer32	double-long:-12345678
double-long-unsigned(6)	Unsigned32	double-long-unsigned:123456
octet-string(9)	Octetstring	octet-string:98aaddff
visible-string(10)	Octetstring	visible-string:31323334
UTF8-string(12)	Octetstring	UTF8-string:31322233
bcd(13)	Unsigned8	bcd:123
integer(15)	Integer8	integer:-4
long(16)	Integer16	long:-12345
unsigned(17)	Unsigned8	unsigned:123
long-unsigned(18)	Unsigned16	long-unsigned:12345
long64(20)	Integer64	long64:123456
long64-unsigned(21)	Unsigned64	long64-unsigned:123456
enum(22)	Integer	enum:3
float32(23)	Octetstring(4)	float32:11223344
float64(24)	Octetstring(8)	float64:1122334455667788
date_time(25)	Octetstring(12)	date_time:00112233445566778899aabb
date(26)	Octetstring(5)	date:1122334455
time(27)	Octetstring(4)	time:00112233
array(1)	[type1:literal1 ,type2:literal2,]	[integer:1,integer:2,integer:3,integer:4]
structure(2)	{ type1:literal1 ,type2:literal2,… }	{boolean:1,octet-string:1234}
compact-array(19)	Type:[literal1, literal2,]	integer:[1,2,3,4]
choice	Anhand des COSEM-Typenamens identifiziert	

Tabelle 13: Mapping von COSEM-Literalen in der URI

7.8. Query Parameter

Query Parameter KÖNNEN dem URI Pfad angehängt werden um bei Lesezugriffen auf Ressourcen die zurückgelieferte Datenmenge nach vorgegebenen Filterkriterien einzuschränken. Dies ist besonders für Listen und Arrays (wie z.B. im Profile Generic Buffer-Attribut) sinnvoll.

Die URI-Query Syntax MUSS in der Form angegeben werden:

<path>?attr$_1$=value$_1${&attr$_n$=value$_n$}

Alle Filter-Parameter die in der Query-URI angegeben sind, sind UND-Verknüpft, d.h. alle Bedingungen MÜSSEN zutreffen.

Tritt ein Query-Parameter mehrmals auf, gilt der Wert seines ersten Auftretens.Weitere Query Parameter gleichen Namens MÜSSEN ignoriert werden.

Ein Query-Parameter für die Dienste Delete, Create, Action MUSS ignoriert werden.

Ein Query-Parameter KANN für den Dienst Get angegeben werden.

Ein Query-Parameter SOLL NICHT für den Dienst Set angegeben werden.

7.8.1. SELEKTIVER ZUGRIFF AUF INHALTE VON COSEM-ATTRIBUTEN

[COSEM] beschreibt einen Selective-Access Mechanismus, mit dem die Anwendung auf einzelne Bestandteile eines Attributes zugreifen kann. Dieser Mechanismus ist in COSEM pro Klasse für ausgewählte Attribute (z.B. Buffer-Attribut einer „Profile-Generic“ Klasse) beschrieben.

Ein Selective-Access ist nur für Get -Zugriffe auf Attribute definiert (nicht für Set, Delete, Create, Action).

Query-Attribute	Bedeutung	Value-Wertebereich	Default Value
sa.fromidx	Gibt den Index des ersten Array Eintrages bei Zugriffen auf COSEM-Array-Attribute an	Integer, 1..n (decimal) or hex (x)	1
sa.count	Limitiert die Zahl der zu übertragenden Einträge bei Read	Integer, 1..n (decimal) or hex leer=Unlimitiert	Leer: Unlimitiert
sa.toidx	Gibt den Index des letzten Eintrages bei Zugriffen auf das adressierte COSEM-Array Attribute an	Integer, 1..n (decimal) or hex leer=Last entry	Leer: Letzer Eintrag
sa.fromcol	Gibt die erste Spalte an, die ausgelesen werden soll	0,1..n	Leer: Erste Spalte
sa.tocol	Gibt die letzte Spalte an, die ausgelesen werden soll	0,1..n	Leer: Letzte Spalte
sa.retrievecolumns	Enthält Liste von Deskriptoren (s.u.) die zur Spaltenauswahl dienen	Descriptor,{Descriptor} leer=Alle Spalten	Leer: Alle
sa.filtercolumn	Enthält den Deskriptor der Spalte auf den die Wertauswahl angewandt wird	Descriptor	
sa.fromtype	Liefert nur Werte die grösser oder gleich (sa.from) sind.	*Type*:value (*Type* nach Kap 7.7)	Leer: Niedrigster Wert
sa.totype	Liefert nur Werte die kleiner oder gleich (sa.to) sind.	*Type*:value (*Type* nach Kap 7.7)	Leer: Höchster Wert

Tabelle 14: Selective-Access Parameter (Entry und Range)

7.8.2. UNIVERSELLE QUERY PARAMETER

Statt streng typisiertem Selective Access kann auch ein COSEM-unabhängiges Query-Schema verwendet werden. Dieses kann für den GET Access die Ressourcen des URI-Baumes nach folgender Tabelle verwendet werden.

Query-Attribute	Bedeutung	Value-Wertebereich	Für Ressourcen	Default Value
q.fromidx	Gibt den Index beim Zugriff auf eine Listen-Ressource an	1..n (decimal), hex (x)	Listen	1
q.count	Limitiert die Zahl der zu übertragenden Einträge bei einer Liste	1..n (decimal), hex(x) leer=Unlimitiert	Listen	Leer: Unlimitiert
q.depth	Legt die Tiefe der Ausgelesen Datenstruktur fest	0..n (decimal), hex (x) 0: Nur das Listen-Element selbst mit Attributen wird geliefert.	Alle	URI mit trailing /: 1 URI ohne trailing /: unlimited
q.fromtime	Filter auf Listen mit Zeitangabe: Ab dieser Zeit	ISO8601	Listen	Kein Limit nach unten
q.totime	Filter auf Listen mit Zeitangabe: Bis zu dieser Zeit	ISO8601	Listen	Kein Limit nach oben

Tabelle 15: Universelle Query Parameter

sa.retrievecolumns/sa.filtercolumns Descriptor:

{<*class*>-<*logical_name*>[,<*attribute_index*>[,<*data_index*>]]}

Die Codierung der Literale entspricht Kap. 7.7

Die Wertebereiche der Indizes gelten entsprechend [COSEM].

Beispiel:
{long:3,octet-string:0100010800ff,integer:2,long:5}

Class 3 – Register
1-0:1.8.0- Energy
Attribute Index 2 – Value
Datenindex 5 – Array Index im Profile-Generic Attribut “buffer”

7.8.3. BILDUNGSREGEL/SYNTAX FÜR SELECTIVE-ACCESS/QUERY-PARAMETER

COSEM-Spezifische Selective Access-Parameter erhalten einen Präfix sa.

Generische Query-Parameter dieser Spezifikation erhalten einen Präfix q.

Die Codierung der Literale entspricht Kap. 7.7

Empfehlung an die Datenmodellierung: Möglichst wenig Gebrauch von Selective Access machen. Modellierung deshalb in Klassen weniger über Array-Attribute, besser mehrere COSEM-Objekte.

7.9. Container-Konzept zum atomaren Zugriff auf aggregierte Ressourcen

Diese Spezifikation führt eine Zusammenfassung von Objekten, Attributen zu einem Container ein. Damit sind atomare Zugriffe auf ein Bündel von Ressourcen möglich. Das Container-Konzept dient auch dazu die Interaktivität (einzelne Zugriffe auf verschiedene Objekte) zu reduzieren.

Der Container ermöglicht den Zugriff auf zusammenhängende Objekte (z.B. ein Regelwerk).

Ein Zugriff auf einzelne Bestandteile des Containers (Objekte, Attribute) über die URI ist nicht vorgesehen.

Container Ressourcen können bereits statisch angelegt sein (Regelwerke, Konfigurationsdatenstrukturen die in atomaren Operationen gelesen/beschrieben werden müssen) bzw. durch das Geräte-Management erweitert werden. Alternativ wird im nächsten Kapitel ein Mechanismus zum Anlegen und Löschen von Ressourcen beschrieben.

Dadurch, dass bestimmte Objekte im URI Baum nicht einzeln sondern nur über einen Container manipulierbar sind, lässt sich eine Atomare Operation auf einem Satz von Klassen erzwingen.

7.10. Listen - Ressourcen

Der URI-Baum enthält Listen-Ressourcen. Der Name einer Listen-Ressource endet mit einem „s“.

Der Name des Elementes einer Listen-Ressource entspricht dem Path-Element Namen in der URI.

Die Liste kann leer sein. Die Anzahl der Listenelemente ist durch diese Spezifikation nicht begrenzt.

Die Unterelemente einer Listen-Ressource besitzen den Namen der Listenressource ohne mit „s“ zu enden.

Listen-Ressourcen sind ldevs, containers, objects, attributes, methods.

Ein GET-Request auf eine Listen-Ressource liefert eine Liste der Elemente unter dieser Ressource. Die Liste kann durch Query-Parameter-Filter eingegrenzt werden (Siehe 7.7).

Die Startposition und Anzahl an Listenelementen wird über Query-Parameter q.fromidx, q.count bestimmt. Dadurch ist ein Blättern durch große Listen möglich.

Die (Verschachtelungs-)Tiefe der zurückgelieferten Element-Strukturen wird durch den q.depth Parameter festgelegt:

Das Listen-Element selbst enthält ein Attribut „count“. Der Wert von “count” enthält die tatsächlich gelieferten Unterelemente.

Das Lesen einer Liste mit einer Tiefe von Null (q.depth=0) liefert nur das obere Listenelement.

```
GET <path>/objects HTTP/1.1
```

liefert

```
<objects count=”2” >
 <object class-id=”3” version=”0” id=”0100010801ff” >
  <attributes>
   <attribute id =”1”>
   ..
   </attribute>
  </attributes>
 </object>
 <object class-id=”3” version=”0” id=”0100010802ff” >
  <attributes>
   <attribute id=”1”>
    ..
   </attribute>
  </attributes>
 </object>
</objects>
```

```
GET <path>/objects/ HTTP/1.1
```

oder

```
GET <path>/objects?q.depth=1 HTTP/1.1
```

liefert

```
<objects count=”2” >
 <object class-id=”3” version=”0” id=”0100010801ff” />
 <object class-id=”3” version=”0” id=”0100010802ff” />
</objects>
```

```
GET <path>/objects?q.depth=0 HTTP/1.1
```

liefert

```
<objects count=”2” />
```

7.11. Dynamisches Anlegen/Löschen von Ressourcen (Containern, Objekten)

Das Datenmodell kann, wenn notwendig und zulässig, über die API dynamisch verändert werden.

Unabhängig von der hier beschriebenen Möglichkeit, die Ressourcen mittels HTTP zu löschen oder anzulegen kann eine Delete/Insert Methode Einträge in Attribut-Arrays löschen/einfügen.

7.11.1. CREATE RESSOURCE

Durch HTTP-PUT an eine Adresse im URI-Baum die noch keine Ressource enthält, wird eine neue Ressource angelegt. Die URI beschreibt die Adresse, die nach einem PUT die neue Ressource enthält.

Hinweis: Ein HTTP-PUT an eine Adresse die bereits eine Ressource enthält überschreibt die vorhandene Ressource.

Das Anlegen einer Ressource per POST an die Adresse einer übergeordneten Ressource ist nicht vorgesehen.

Beispiel:

Client → Server:

```
PUT <PoC>/cosem/ldevs/<ldevid>

{Body}
XML-Daten der Objekt-Instanz.
```

Server → Client:

```
HTTP/1.1 201 Created
```

Legt ein neues Logical_Device mit der ID <ldevid> an. Die initialen Daten (COSEM-Objekte) werden im HTTP-Body übergeben.

Falls kein HTTP-Body vorhanden ist, wird die Ressource leer angelegt.

7.11.2. DELETE RESOURCE

Durch HTTP-DELETE wird eine Ressource gelöscht.

Client → Server:

```
DELETE <PoC>/cosem/ldevs/<ldevid>
```

Server → Client:

```
HTTP/1.1 204 No content
```

7.12. Fragmentierung von Inhaltsdaten bei Abbruch von Transport-Verbindungen

Nach [HTTP] KÖNNEN Clients Idempotente Requests (GET, PUT, DELETE) wiederholen, falls die Transport-Verbindung und damit die http-Session beendet wurde und die Response noch nicht (komplett) durch den Client empfangen wurde. Abbrüche der Transport-Verbindung während der Ausführung von nicht-Idempotenten Requests (POST) MÜSSEN der Anwendung mitgeteilt werden.

Hinweis: Grosse HTTP-Requests/Responses können zum einen in unter http liegenden Transportschichten fragmentiert und reassembliert werden. Diese Fragmentierung ist für den http-Layer und die Anwendung transparent.

Wird die unterliegende Transportverbindung getrennt muss auch die http-Verbindung getrennt. Ein häufiges Trennen der http-Verbindung mit Wiederholen des Requests führt zu erhöhtem Datenvolumen. Der durch HTTP angebotene Blocktransfer ermöglicht eine Fragmentierung auf http-Ebene.

Eine Fragmentierung auf Anwendungsebene ist über COSEM-Klassen zu realisieren. Z.B. ermöglicht die COSEM Image-Transfer-Klasse einen auf Anwendungsebene gesteuerten Blocktransfer.

Bei Listen undAttributen ist eine Teilweise Übertragung (Blättern oder „Paging“) auf Anwendungsebene über die Verwendung von Selective Access bzw. Query Parameter möglich. Dieser Mechanismus kann für das Lesen von Daten nicht aber für die Zustellung von Daten verwendet werden.

7.12.1. Übertragen von grossen HTTP-Bodies durch Blocktransfer in http

http-Blocktransfer ist nur für Idempotente Operationen (PUT, GET) zugelassen. Die Operation gilt erst nach der Übertragung des letzten Blockes als abgeschlossen.

Hinweis: Es wird angenommen, dass der Server die Ressource zu Beginn eines durch den Client eingeleiteten Blocktransfers in einen Zwischenpuffer kopiert.

Der Client legt im Request fest, dass er nur einen Teil einer Ressource lesen möchte.

Beispiel Lesen einer Ressource von 1000 Bytes in zwei Blöcken:

Request:

```
GET /largecontent http/1.0
Host: <host>
Range: bytes 0-511
```

Hinweis: Der Server kann die Ressource hier in einen Zwischenspeicher kopieren.

Response:

```
HTTP/1.1 206 Partial Content
Content-range: bytes 0-511/1000
Content-length: 512
{ Data }
```

Request:

```
GET /largecontent http/1.0
Host: <host>
Range: bytes 512-
```

Response:

```
HTTP/1.1 206 Partial Content
Content-range: bytes 512-999/1000
Content-length: 488
{ Data }
```

Hinweis: Der Server kann den Zwischenspeicher hier freigeben.

8. Inhaltsdaten (http Content-Body)

8.1. Hinweis zum Zugriffslayer

Diese Spezifikation beschreibt einen Vorschlag, bei dem der Content-Body die Datenstrukturen um den COSEM-ASN1 Typ „Data“ enthält. Der COSEM-Access (wie beispielsweise in IEC62056-53) ist nicht Bestandteil des Content-Bodies, da er durch [HTTP] realisiert wird.

8.2. URI Resource-Tree eines Physical COSEM Devices

8.2.1. Der Point-Of-Contact (<PoC>)

Der Point-Of-Contact ist die Wurzel des Ressourcen Baumes dieser Spezifikation. Anders ausgedrückt ist der Point-Of-Contact der URI-Path-Prefix für die API dieser Spezifikation.

Die Wurzel des COSEM-Baumes <PoC> MUSS der Server dem Client zuvor bekannt gemacht haben.

8.2.2. XML Inhalts-Codierung

Der XML Inhalt MUSS in UTF-8 codiert werden:

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
```

8.2.3. XSD Schema Relation

Das Schema für die COSEM Datentypen [XSD-COD] wird benötigt und eingebunden.

Beispiel:

```
<xsd:schema
        xmlns:cod="urn:k461-dke-de:cod-1“
        attributeeFormDefault="unqualified"
        elementFormDefault="qualified"
        xmlns:cor=”urn:k461-dke-de:cor-1”
        targetNamespace="urn:k461-dke-de:cor-1”
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Normativ gilt das Schema in den Dateien [XSD-COD] und [XSD-COR].

8.2.4. COSEM Element – Wurzel des COSEM Physical Device

Zulässige Operationen auf der Ressource

GET <poc>/cosem – Liefert COSEM-Element incl. Unterlemente (bei entsprechender Berechtigung)

Schema:

```
<xsd:complexType name="cosemType">
<xsd:sequence>
<xsd:element name="ldevs" type="ldevsType" minOccurs="1" maxOccurs="unbounded" />
<xsd:element name="containers" type="containersType" minOccurs="0"
maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

<xsd:element name="cosem" type="cosemType" />
```

Beispiel:

```
<cosem>
<ldevs count="1">
….
</ldevs>
<containers count="1">
….
</containers>
</cosem>
```

Das cosem-Element enthält immer ein ldevs-Element (Liste) in dem die Liste der Logical-Devices angelegt ist.

Das Container Element ist optional. Container auf dieser Ebene bündeln mehrere Logical Devices.

8.2.5. LVDEVS ELEMENT – LISTE DER LOGICAL DEVICES

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs – Liefert Liste der Logical Devices incl. Unterlementen (bei entsprechender Berechtigung)

Schema:

```
<xsd:complexType name="ldevsType">
<xsd:sequence>
<xsd:element name="ldev" type="ldevType" minOccurs="1" maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="count" type="xsd:unsignedInt" use="optional" />
</xsd:complexType>

<xsd:element name="ldevs" type="ldevsType" />
```

Beispiel:

```
<ldevs count="1">
<ldev id="1din0a12345678.sm[.<domain>]" />
</ldevs>
```

Ein ldevs-Element muss nach IEC62056-6-2 ein Management LogicalDevice für das Phyiscal Device besitzen.

8.2.6. LDEV ELEMENT –WURZEL DES LOGICAL DEVICE

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid> – Liefert den Inhalt (die Objekte) eines Logical Device als Response Body

PUT <poc>/cosem/ldevs/<ldevid> – Erzeugt dynamisch ein neues Logical Device (ggf. mit Objekten) mit dem Inhalt des Request-Bodies. Falls das Logical Device bereits existiert werden Unterelemente/Objekte neu angelegt oder überschrieben.

DELETE <poc>/cosem/ldevs/<ldevid> – Löscht dynamisch ein Logical Device (ggf. mit Objekten)
Hinweis: Das Management Logical Device kann nicht gelöscht werden.

Schema:

```
<xsd:complexType name="ldevType">
<xsd:sequence>
<xsd:element name="objects type="objectsType" minOccurs="1" maxOccurs="unbounded"
/>
<xsd:element name="containers" type="ldevContainersType" minOccurs="0"
maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="id" type="cod:octet-string" use="required" />
</xsd:complexType>

<xsd:element name="ldev" type="ldevType" />
```

Beispiel:

```
<ldev id="1din0a12345678.sm[.domain]" >
<objects count="1" >
…
</objects>
</ldev>
```

8.2.7. OBJECTS ELEMENT –LISTE ALLER OBJEKTE EINES LOGICAL DEVICE

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/objects – Liefert die Objekte eines Logical Device als Response-Body

Schema:

```
<xsd:complexType name="objectsType">
<xsd:sequence>
<xsd:element name="object" type="objectType" />
</xsd:sequence>
<xsd:attribute name="count" type="xsd:unsignedInt" use="optional"/>
</xsd:complexType>

<xsd:element name="objects" type="objectsType" />
```

Beispiel:

```
<objects count="2" >
<object id="0100010801ff" class-id="3" version="0" />
```

```
<object id=”0100010802ff” class-id=”3” version=”0” />
</objects>
```

8.2.8. OBJECT ELEMENT –COSEM OBJEKT

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/objects/<classid-objid> – Liefert den Inhalt eines Objektes eines LogicalDevice als Response-Body

PUT <poc>/cosem/ldevs/<ldevid>/objects/<classid-objid> – Erzeugt dynamisch oder überschreibt den Inhalt eines Objektes eines LogicalDevice mit dem Inhalt des Request-Bodies

DELETE <poc>/cosem/ldevs/<ldevid>/objects/<classid-objid> – Löscht das adressierte Objekt.

Schema:

```
<xsd:complexType name=”objectType”>
<xsd:sequence>
<xsd:element name=”attributes” type=”attributesType” minOccurs=”1”
maxOccurs=”unbounded” />
<xsd:element name=”methods” type=”methodsType” minOccurs=”0” maxOccurs=”unbounded”
/>
</xsd:sequence>
<xsd:attribute name="id" type="cod:Cosem-Object-Instance-Id" use="required"/>
<xsd:attribute name="class-id" type="cod:Cosem-Class-Id" use="required"/>
<xsd:attribute name="version" type="Unsigned8" use="required"/>
</xsd:complexType>

<xsd:element name=”object” type=”objectType” />
```

Beispiel:

```
<object class-id=”3” version=”0” id=“0100010800ff”>
<attributes count=”1” >
(Data)
</attributes>
<methods count=”1” >
</methods>
</object>
```

8.2.9. ATTRIBUTES ELEMENT –LISTE ALLER ATTRIBUTE EINES COSEM OBJEKTES

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/objects/<classid-objid>/attributes – Liefert alle Attribute eines Objektes als Response-Body

Schema:

```
<xsd:complexType name=”attributesType”>
<xsd:sequence>
```

```
<xsd:element name=”attribute” type=”attributeType” minOccurs=”1”
maxOccurs=”unbounded” />
</xsd:sequence>
<xsd:attribute name=”count” type=”xsd:unsignedInt” use=”optional”/>
</xsd:complexType>


<xsd:element name=”attributes” type=”attributesType” />
```

Beispiel:

```
<attributes count=”2”>
<attribute id=”1” >
<octet-string>0100010800ff</octet-string>
</attribute>
<attribute id=”2” >
<Integer8>1</Integer8>
</attribute>
</attributes>
```

8.2.10. ATTRIBUTE ELEMENT –COSEM ATTRIBUT

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/objects/<objid>/attributes/<attrid> – Liefert den Inhalt eines Attributes eines Objektes als Response-Body

PUT <poc>/cosem/ldevs/<ldevid>/objects/<objid>/attributes/<attrid> – Überschreibt den Inhalt eines Attributes mit dem Inhalt des Request-Bodies

Schema:

```
<xsd:complexType name=”attributeType”>
<xsd:sequence>
<xsd:element name=”attribute” type=”cod:Data” min_occurs=”1”
max_occurs=”1” />
</xsd:sequence>
<xsd:attribute name=”id” type=”cod:Cosem-Attribute-Id-Type”
use=”required” />
</xsd:complexType>
```

Beispiel:

```
<attribute id=”1” >
<octet-string>0100010801ff</octet-string>
</attribute>
```

8.2.11. METHODS ELEMENT –LISTE ALLER METHODEN EINES COSEM OBJEKTES

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/objects/<class-objid>/methods – Liefert Liste aller Methoden eines Objektes als Response-Body

Schema:

```
<xsd:complexType name=”methodsType”>
<xsd:sequence>
<xsd:element name=”method” type=”methodType” min_occurs=”0”
max_occurs=”unbounded” />
</xsd:sequence>
<xsd:attribute name=”count” type=”xsd:unsignedInt” use=”optional”/>
</xsd:complexType>
```

Beispiel:

```
<methods count=”2” “>
<method id=”1” />
<method id=”2” />
</methods>
```

8.2.12. METHOD ELEMENT –COSEM METHODE

Input-Parameter und Output-Parameter sind vom Typ cod:Data.

Zulässige Operationen auf der Ressource

POST <poc>/cosem/ldevs/<ldevid>/objects/<objid>/methods/<methodid> – Ruft die Methode mit den Daten des Request-Bodies auf. Ein Fehler-Code wird im Response-Body übergeben. (http-Statuscode 500) Evtl. Output-Parameter werden im Response-Body zurückgeliefert.

Schema:

Wird als Response auf GET <…>/methods Listen-Ressource geliefert

```
<xsd:complexType name=”methodType”>
<xsd:sequence />
<xsd:attribute name=”id” type=”cod:Cosem-Method-Id-Type”
use=”required” />
</xsd:complexType>
```

Wird als Request Body für POST <…>/methods/<id> verwendet:

```
<xsd:element name=”methodRequestParameter” type=”cod:Data”
min_occurs=”0” max_occurs=”1” />
```

Wird als Response auf POST <…>/methods/<id> geliefert

```
<xsd:complexType name=”methodResponseType”>
<xsd:sequence>
<xsd:element name=”methodResponseStatus” type=”cod:Unsigned8”
min_occurs=”0” max_occurs=”1” />
<xsd:element name=”methodResponseData” type=”cod:Data”
min_occurs=”0” max_occurs=”1” />
</xsd:sequence>
</xsd:complexType>


<xsd:element name=”methodResponse” type=”methodResponseType” />
```

Beispiel Request-Body:

```
…
<methodRequestParameter>
<Integer8>0</Integer8>
</methodRequestParameter>
…
```

Beispiel Response-Body:

```
…
<methodResponse>
<methodResponseData>
<Integer8>0</Integer8>
</methodResponseData>
</methodRequest>
…
```

Ein methodResponseStatus=0 wird mit HTTP-Statuscode 200 übermittelt. Ein methodResponseStatus >0 wird mit http-Statuscode 500 übermittelt.

8.2.13. CONTAINERS ELEMENT –LISTE ALLER CONTAINER EINES LOGICAL DEVICES

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/containers – Liefert Liste aller Container eines LogicalDevices als Response-Body

Schema:

```
<xsd:complexType name=”ldevContainersType”>
<xsd:sequence>
<xsd:element name=”container” type=”ldevContainerType”
min_occurs=”0” max_occurs=”unbounded” />
</xsd:sequence>
<xsd:attribute name=”count” type=”xsd:unsignedInt” use=”optional”/>
</xsd:complexType>

<xsd:element name=”containers” type=”ldevContainersType” >
```

GET/PUT Body Beispiel:

```
<containers count=”1”>
<container id=”00005e3101ff”>
<object class-id=”3” version=”0” id=”0100010801ff”>
…
</object>
<object class-id=”3” version=”0” id=”0100010802ff”>
…
</object>
</container>
</containers>
```

8.2.14. CONTAINER ELEMENT IM LOGICAL DEVICE

Zulässige Operationen auf der Ressource

GET <poc>/cosem/ldevs/<ldevid>/containers/<containerid> – Liefert den Inhalt eines Containers als Response-Body

PUT <poc>/cosem/ldevs/<ldevid>/containers/<containerid> – Erzeugt oder überschreibt den Inhalt eines Containers mit dem Inhalt des Request-Bodies

DELETE <poc>/cosem/ldevs/<ldevid>/containers/<containerid> – Entfernt einen Container

```
<xsd:complexType name=”ldevContainerType”>
<xsd:sequence>
<xsd:element name=”container” type=”objectsType” />
</xsd:sequence>
<xsd:attribute name=”id” type=”cod:Cosem-Object-Instance-Id” />
</xsd:complexType>

<xsd:element name=”container” type=”ldevContainerType” >
```

GET/PUT Body Beispiel:

```
<containers count=”1” “>
<container id=”00005e3101ff”>
<object class-id=”3” version=”0” id=”0100010801ff”>
…
</object>
<object class-id=”3” version=”0” id=”0100010802ff”>
…
</object>
</container>
</containers>
```

8.2.15. CONTAINERS ELEMENT –LISTE ALLER CONTAINER

Zulässige Operationen auf der Ressource

GET <poc>/cosem/containers – Liefert Liste aller Container als Response-Body

Schema:

```
<xsd:complexType name=”containersType”>
<xsd:sequence>
<xsd:element name=”container” type=”containerType” minOccurs=”0”
maxOccurs=”unbounded” />
</xsd:sequence>
<xsd:attribute name=”count” type=”xsd:unsignedInt” use=”optional”/>
</xsd:complexType>
```

8.2.16. CONTAINER ELEMENT

Zulässige Operationen auf der Ressource

GET <poc>/cosem/containers/<containerid> – Liefert den Inhalt eines Containers als Response-Body

PUT <poc>/cosem/containers/<containerid> – Erzeugt oder überschreibt den Inhalt eines Containers mit dem Inhalt des Request-Bodies

DELETE <poc>/cosem/containers/<containerid> – Entfernt einen Container

```
<xsd:complexType name=”containerType”>
<xsd:sequence>
<xsd:element name=”container” type=”ldevsType” />
</xsd:sequence>
<xsd:attribute name=”id” type=”hexBinary” />
</xsd:complexType>
```

8.3. Datentypen für COSEM-Objekte und Attribute

Die Common Datatypes aus IEC62056-6-2:2010 Kap 4.5 werden unterstützt.

```
<?xml version="1.0" encoding="UTF-8" ?>

 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:k461-dke-
de:cod-1 -targetNamespace=“urn:k461-dke-de:cod-1“ elementFormDefault="qualified">

 <xsd:simpleType name="Integer8">
<xsd:restriction base="xsd:byte" />
</xsd:simpleType>

<xsd:simpleType name="Integer16">
<xsd:restriction base="xsd:short" />
</xsd:simpleType>

 <xsd:simpleType name="Integer32">
<xsd:restriction base="xsd:int" />
</xsd:simpleType>

 <xsd:simpleType name="Integer64">
<xsd:restriction base="xsd:long" />
</xsd:simpleType>

<xsd:simpleType name="Unsigned8">
<xsd:restriction base="xsd:unsignedByte" />
</xsd:simpleType>

 <xsd:simpleType name="Unsigned16">
<xsd:restriction base="xsd:unsignedShort" />
</xsd:simpleType>
```

```
<xsd:simpleType name="Unsigned32">
<xsd:restriction base="xsd:unsignedInt" />
</xsd:simpleType>

<xsd:simpleType name="Unsigned64">
<xsd:restriction base="xsd:unsignedLong" />
</xsd:simpleType>

<xsd:simpleType name="Cosem-Class-Id">
<xsd:restriction base="Unsigned16" />
</xsd:simpleType>

<xsd:simpleType name="Cosem-Object-Instance-Id">
<xsd:restriction base="xsd:hexBinary" >
</xsd:simpleType>

<xsd:simpleType name="Cosem-Object-Attribute-Id">
<xsd:restriction base="Integer8" />
</xsd:simpleType>

<xsd:simpleType name="Cosem-Object-Method-Id">
<xsd:restriction base="Integer8" />
</xsd:simpleType>

<xsd:complexType name="NULL" final="#all" />

<xsd:simpleType name="BitString">
<xsd:restriction base="xsd:string">
<xsd:pattern value="[0-1]{0,}" />
</xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="TypeDescription">
<xsd:choice>

<xsd:element name="null-data" type="NULL" />

<xsd:element name="array">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="number-of-elements" type="Unsigned16" />
<xsd:element name="type-description" type="TypeDescription" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="structure">
<xsd:complexType>
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<xsd:element name="TypeDescription" type="TypeDescription" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

```
<xsd:element name="boolean" type="NULL" />

<xsd:element name="bit-string" type="NULL" />

<xsd:element name="double-long" type="NULL" />

<xsd:element name="double-long-unsigned" type="NULL" />

<xsd:element name="floating-point" type="NULL" />

<xsd:element name="octet-string" type="NULL" />

<xsd:element name="visible-string" type="NULL" />

<xsd:element name="UTF8-string" type="NULL" />

<xsd:element name="bcd" type="NULL" />

<xsd:element name="integer" type="NULL" />

<xsd:element name="long" type="NULL" />

<xsd:element name="unsigned" type="NULL" />

<xsd:element name="long-unsigned" type="NULL" />

<xsd:element name="long64" type="NULL" />

<xsd:element name="long64-unsigned" type="NULL" />

<xsd:element name="enum" type="NULL" />

<xsd:element name="float32" type="NULL" />

<xsd:element name="float64" type="NULL" />

<xsd:element name="date_time" type="NULL" />

<xsd:element name="date" type="#NULL" />

<xsd:element name="time" type="NULL" />

<xsd:element name="dont-care" type="NULL" />

</xsd:choice>
</xsd:complexType>



 <xsd:complexType name="SequenceOfData">
<xsd:choice minOccurs="0" maxOccurs="unbounded">

<xsd:element name="null-data" type="NULL" />

```

```
<xsd:element name="array" type="SequenceOfData" />

<xsd:element name="structure" type="SequenceOfData" />

<xsd:element name="boolean" type="xsd:boolean" />

<xsd:element name="bit-string" type="BitString" />

<xsd:element name="double-long" type="Integer32" />

<xsd:element name="double-long-unsigned" type="Unsigned32" />

 <xsd:element name="floating-point">
 <xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
<xsd:length value="4" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="octet-string" type="xsd:hexBinary" />

<xsd:element name="visible-string" type="xsd:string" />

<xsd:element name="UTF8-string" type="xsd:string" />

<xsd:element name="bcd" type="Integer8" />

<xsd:element name="integer" type="Integer8" />

<xsd:element name="long" type="Integer16" />

<xsd:element name="unsigned" type="Unsigned8" />

<xsd:element name="long-unsigned" type="Unsigned16" />

<xsd:element name="compact-array">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="contents-description" type="TypeDescription" />
<xsd:element name="array-contents" type="xsd:hexBinary" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="long64" type="Integer64" />

<xsd:element name="long64-unsigned" type="Unsigned64" />

<xsd:element name="enum" type="Unsigned8" />

<xsd:element name="float32">

<xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
```

```
<xsd:length value="4" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

 <xsd:element name="float64">
 <xsd:simpleType>
 <xsd:restriction base="xsd:hexBinary">
<xsd:length value="8" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

 <xsd:element name="date_time">
 <xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
<xsd:length value="12" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="date">
<xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
<xsd:length value="5" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

 <xsd:element name="time">
<xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
<xsd:length value="4" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="dont-care" type="NULL" />

</xsd:choice>
</xsd:complexType>



 <xsd:complexType name="Data">
 <xsd:choice>

<xsd:element name="null-data" type="NULL" />

<xsd:element name="array" type="SequenceOfData" />

<xsd:element name="structure" type="SequenceOfData" />

<xsd:element name="boolean" type="xsd:boolean" />
```

```
<xsd:element name="bit-string" type="BitString" />

<xsd:element name="double-long" type="Integer32" />

<xsd:element name="double-long-unsigned" type="Unsigned32" />

<xsd:element name="floating-point">
 <xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
<xsd:length value="4" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="octet-string" type="xsd:hexBinary" />

<xsd:element name="visible-string" type="xsd:string" />

<xsd:element name="UTF8-string" type="xsd:string" />

<xsd:element name="bcd" type="Integer8" />

<xsd:element name="integer" type="Integer8" />

<xsd:element name="long" type="Integer16" />

<xsd:element name="unsigned" type="Unsigned8" />

<xsd:element name="long-unsigned" type="Unsigned16" />

 <xsd:element name="compact-array">
 <xsd:complexType>
 <xsd:sequence>
<xsd:element name="contents-description" type="TypeDescription" />
<xsd:element name="array-contents" type="xsd:hexBinary" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="long64" type="Integer64" />

<xsd:element name="long64-unsigned" type="Unsigned64" />

<xsd:element name="enum" type="Unsigned8" />

<xsd:element name="float32">
<xsd:simpleType>
 <xsd:restriction base="xsd:hexBinary">
<xsd:length value="4" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="float64">
 <xsd:simpleType>
```

```
 <xsd:restriction base="xsd:hexBinary">
<xsd:length value="8" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="date_time">
 <xsd:simpleType>
 <xsd:restriction base="xsd:hexBinary">
<xsd:length value="12" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

 <xsd:element name="date">
 <xsd:simpleType>
 <xsd:restriction base="xsd:hexBinary">
<xsd:length value="5" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="time">
<xsd:simpleType>
<xsd:restriction base="xsd:hexBinary">
<xsd:length value="4" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="dont-care" type="NULL" />
</xsd:choice>
</xsd:complexType>

</xsd:schema>
```