



1 **Technische Richtlinie BSI TR-03109-1**

2 **Anlage II: COSEM/HTTP Webservices**

3

4 Version 1.0, Datum 18.03.2012

5

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>

© Bundesamt für Sicherheit in der Informationstechnik 2013

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

7			
8			
9			
10			
11			
12			
13	1.	DOKUMENTENHISTORIE	5
14	2.	REFERENZIERTER DOKUMENTE	6
15	3.	ABKÜRZUNGEN UND BEGRIFFE	7
16	4.	EINLEITUNG	8
17	4.1.	SCOPE	8
18	4.2.	DOKUMENTENSTRUKTUR	8
19	4.3.	ANWENDUNGSBEREICH	9
20	4.4.	ÜBERSICHT ÜBER WESENTLICHE ANFORDERUNGEN AN DIE SPEZIFIKATION	9
21	5.	DATENMODELL UND API-STRUKTUR	10
22	5.1.	COSEM PHYSICAL DEVICE, LOGICAL DEVICE, OBJECT, ATTRIBUTES, METHODS,	10
23	5.2.	URI RESOURCE-TREE EINES PHYSICAL COSEM DEVICES	11
24	6.	ABBILDUNG VON IDENTIFIKATOREN UND BEZEICHNERN	12
25	6.1.	KANONISCHER GERÄTEBEZEICHNER	12
26	6.2.	HOSTNAME	12
27	6.3.	LOGICAL_DEVICEID	12
28	6.4.	COSEM-OBJEKT-BEZEICHNER IN DER URI	13
29	6.5.	COSEM ATTRIBUT-INDEX	13
30	6.6.	COSEM METHODEN-INDEX	13
31	6.7.	CONTAINER BEZEICHNER IN DER URI	13
32	7.	COSEM ZUGRIFF ÜBER HTTP	14
33	7.1.	HTTP-REQUEST/RESPONSE	14
34	7.1.1.	<i>HTTP Response Timeout</i>	15
35	7.1.2.	<i>Nutzung von mehr als einer Verbindung zwischen Kommunikationspartnern</i>	15
36	7.2.	ZUORDNUNG VON ZUGRIFFS-DIENSTEN UND HTTP-VERBEN	15
37	7.3.	HTTP-HEADER	16
38	7.3.1.	<i>Request-Header</i>	16
39	7.3.1.1.	Content-Type Header	17
40	7.3.1.2.	Content-Encoding Header	18
41	7.3.1.3.	Content-Length Header	18
42	7.3.1.4.	Host Header	18
43	7.3.1.5.	Accept Header	18
44	7.3.1.6.	Accept-Encoding Header	18
45	7.3.1.7.	x-CID Header	18
46	7.3.1.8.	Range Header	19

47	7.3.1.9.	x-ContactURI Header	19
48	7.3.1.10.	HTTP Kompression	19
49	7.3.2.	<i>Response-Header</i>	19
50	7.3.2.1.	Content-Type Header	20
51	7.3.2.2.	Content-Encoding Header	20
52	7.3.2.3.	Content-Length Header	20
53	7.3.2.4.	Content-Range Header	20
54	7.3.2.5.	Retry-after Header	20
55	7.4.	HTTP STATUS-CODES	21
56	7.5.	SYNCHRONE/ASYNCHRONE ANTWORT	22
57	7.6.	ZUGRIFFSRECHTE	23
58	7.7.	REPRÄSENTATION VON COSEM LITERALEN IN DER HTTP-ZUGRIFFSSCHICHT	23
59	7.8.	QUERY PARAMETER	24
60	7.8.1.	<i>Selektiver Zugriff auf Inhalte von COSEM-Attributen</i>	25
61	7.8.2.	<i>Universelle Query Parameter</i>	26
62	7.8.3.	<i>Bildungsregel/Syntax für Selective-Access/Query-Parameter</i>	26
63	7.9.	CONTAINER-KONZEPT ZUM ATOMAREN ZUGRIFF AUF AGGREGIERTE RESSOURCEN	27
64	7.10.	LISTEN - RESSOURCEN	27
65	7.11.	DYNAMISCHES ANLEGEN/LÖSCHEN VON RESSOURCEN (CONTAINERN, OBJEKTEN)	29
66	7.11.1.	<i>Create Ressource</i>	29
67	7.11.2.	<i>Delete Resource</i>	29
68	7.12.	FRAGMENTIERUNG VON INHALTSDATEN BEI ABBRUCH VON TRANSPORT-VERBINDUNGEN	30
69	7.12.1.	<i>Übertragen von grossen HTTP-Bodies durch Blocktransfer in http</i>	30
70	8.	INHALTSDATEN (HTTP CONTENT-BODY)	32
71	8.1.	HINWEIS ZUM ZUGRIFFSLAYER	32
72	8.2.	URI RESOURCE-TREE EINES PHYSICAL COSEM DEVICES	32
73	8.2.1.	<i>Der Point-Of-Contact (<PoC>)</i>	32
74	8.2.2.	<i>XML Inhalts-Codierung</i>	32
75	8.2.3.	<i>XSD Schema Relation</i>	32
76	8.2.4.	<i>cosem Element – Wurzel des COSEM Physical Device</i>	32
77	8.2.5.	<i>lvdevs Element – Liste der Logical Devices</i>	33
78	8.2.6.	<i>ldev Element –Wurzel des Logical Device</i>	34
79	8.2.7.	<i>objects Element –Liste aller Objekte eines Logical Device</i>	34
80	8.2.8.	<i>object Element –COSEM Objekt</i>	35
81	8.2.9.	<i>attributes Element –Liste aller Attribute eines COSEM Objektes</i>	35
82	8.2.10.	<i>attribute Element –COSEM Attribut</i>	36
83	8.2.11.	<i>methods Element –Liste aller Methoden eines COSEM Objektes</i>	36
84	8.2.12.	<i>method Element –COSEM Methode</i>	37
85	8.2.13.	<i>Containers Element –Liste aller Container eines Logical Devices</i>	38
86	8.2.14.	<i>Container Element im Logical Device</i>	38
87	8.2.15.	<i>Containers Element –Liste aller Container</i>	39
88	8.2.16.	<i>Container Element</i>	39
89	8.3.	DATENTYPEN FÜR COSEM-OBJEKTE UND ATTRIBUTE	40
90			

Tabellen

91	
92	TABELLE 1: LISTE REFERENZierter DOKUMENTE..... 7
93	TABELLE 2: EINORDNUNG IN OSI LAYER 8
94	TABELLE 3: POSTFIX FÜR NUMMERIERUNGSSHEMA 12
95	TABELLE 4: ATTRIBUT INDIZES 13
96	TABELLE 5: ZUGRIFFS-DIENSTE 16
97	TABELLE 6: IDEMPOTENZ UND SAFENESS VON HTTP-VERBEN..... 16
98	TABELLE 7: RESPONSE HEADER..... 17
99	TABELLE 8: CONTENT-TYPES 17
100	TABELLE 9: CONTENT-TYPE PARAMETER..... 17
101	TABELLE 8: CONTENT-ENCODINGS FÜR CONTENT-ENCODING/ACCEPT-ENCODING 18
102	TABELLE 11: RESPONSE-HEADER 20
103	TABELLE 11: HTTP-STATUSCODES 21
104	TABELLE 12: MAPPING VON COSEM-LITERALEN IN DER URI..... 24
105	TABELLE 13: SELECTIVE-ACCESS PARAMETER (ENTRY UND RANGE) 25
106	TABELLE 14: UNIVERSELLE QUERY PARAMETER 26

107

Bilder

108	
109	ABBILDUNG 1: STRUKTUR DES COSEM PHYSICAL DEVICE..... 10
110	ABBILDUNG 2: URI BAUM..... 11

111

1. Dokumentenhistorie

113

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,RPI,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,RPI,HBa,MSt

		„TLS“ separiert.	
22.11.2012	0.80	Redaktionskreis: TR03109 Bezug in Anhang (Kap 9). Keine Referenzen auf TR03109 im Hauptteil.	MCa,RPI,MSt
23.11.2012	0.81	Gerätekenzeichnung 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,RPI,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

114

115 2. Referenzierte Dokumente

116 Folgende Schriftstücke werden durch dieses Dokument adressiert:

117

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

118 **Tabelle 1: Liste referenzierter Dokumente**119 **3. Abkürzungen und Begriffe**

120 Abkürzungen und Begriffe werden wie folgt benutzt:

121 APDU – Application Protocol Data Unit (Inhaltsdatenstruktur einer Nachricht)

122 Asset – Im Kontext des Schutzprofiles ein schützenswertes Gut (z.B. eine Datenstruktur oder
123 ein Speicher (-inhalt))124 Body /Content-Body – Ein HTTP Request und Response teilt sich in einen Header (Kopf)
125 und einen Body (Rumpf). Der Body enthält den zu transportierenden Inhalt der Nachricht.126 COSEM - Companion Specification for Energy Metering (Internationale Spezifikation zur
127 beschreibt von Datenstrukturen/Klassen für das Messwesen)

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

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

131 EMT – Externer Marktteilnehmer

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

133 HES – Head End System (Terminiert die Kommunikationsverbindungen auf den Transport
134 oder TLS-Layer). Betrieben von Kommunikationsdienstleistern (und Externen
135 Marktteilnehmern).136 HMAC – Hashed Message Authentication Code (eine mittels Geheimnis gebildete
137 Prüfsumme)

138 HTTP – Hypertext Transfer Protocol

139 Objekt – Instanz einer Klasse. Ein COSEM-Objekt enthält COSEM-Attribute und COSEM-
140 Methoden.141 Ressource – Bezeichnet im HTTP/REST Kontext ein über den URI-Baum adressierbares
142 Element. Dies kann ein Container oder im COSEM Kontext ein Physical_Device,
143 Logical_Device, Object, Attribut oder eine Methode sein.

- 144 Request – Die Anforderungsnachricht vom Client zum Server. Wird vom Server mit einer
145 Response beantwortet.
- 146 Response – Die Antwortnachricht vom Server zum Client. Wird beim Client verarbeitet.
- 147 REST(ful) – Representational State Transfer (Stil eine API und einen Datenzugriff über
148 Ressourcen mit einheitlichem CRUD-Model abzubilden)
- 149 RFC – Request for comments. IETF Spezifikation oder Standard.
- 150 SMGW – Smart Meter Gateway
- 151 SMGW_ADM – Smart Meter Gateway Administrator
- 152 TR - Technische Richtlinie
- 153 Verb – Im Kontext von HTTP eine Zugriffsmethode: GET, PUT, POST, DELETE

154 4. Einleitung

155 4.1. Scope

156 Dieses Dokument beschreibt:

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

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

167 4.2. Dokumentenstruktur

168 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		

169 **Tabelle 2: Einordnung in OSI Layer**

170 Der Grüne Bereich wird in dieser Spezifikation beschrieben.

171

172 4.3. Anwendungsbereich

173 Die vorliegende Spezifikation beschreibt die Abbildung der in den Standards [COSEM]
174 (COSEM-Interface-Classes) und [OBIS] bereitgestellten Definitionen über HTTP mittels
175 eines RESTful API Stiles.
176

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

180

181 Die Spezifikation ist im Protokollstack alternativ zu IEC62056-53 angeordnet.
182

183 4.4. Übersicht über wesentliche Anforderungen an die Spezifikation

184

185 1. HTTP/1.1 wird als Webservice-Protokoll verwendet.

186 2. Abbildung von COSEM-Interface-Klassen über HTTP. Zusätzlich kann über die
187 Webservice-Schnittstelle Administration (z.B. File-Download) und Mehrwertdienste
188 auch ohne COSEM-Klassen-Abbildung genutzt werden (dies ist nicht mehr Teil dieser
189 Spezifikation).

190 3. Abbildung des COSEM-IC Zugriffes über RESTful Verben und URI-Adressierung

191 4. Der Inhalt des HTTP-Body ist abhängig vom Content-Type/Accepted und soll auch
192 Inhaltsdatengesicherter Transport ermöglichen. Der Client kann dem Server mitteilen,
193 welche Inhaltsdatentypen er akzeptiert.

194 5. Ermöglichen der für den Betrieb von intelligenten Messsystemen notwendigen UseCases
195 und den daraus abgeleiteten Kommunikationsszenarien: MANAGEMENT, ADMIN-
196 SERVICE, INFO-REPORT.

197 6. Einschränkung der Zugriffsrechte auf Server-Ressourcen abhängig von der
198 Authentifizierung des Clients.

199 7. Auftrennung der Client/Server und Server/Client Funktion auf HTTP-Ebene zwischen
200 Gateway und Administrator in zwei parallele Verbindungen.

201 • Die HTTP-Session wird zusammen mit der darunterliegenden (sicheren)
202 Transportverbindung auf- und abgebaut.

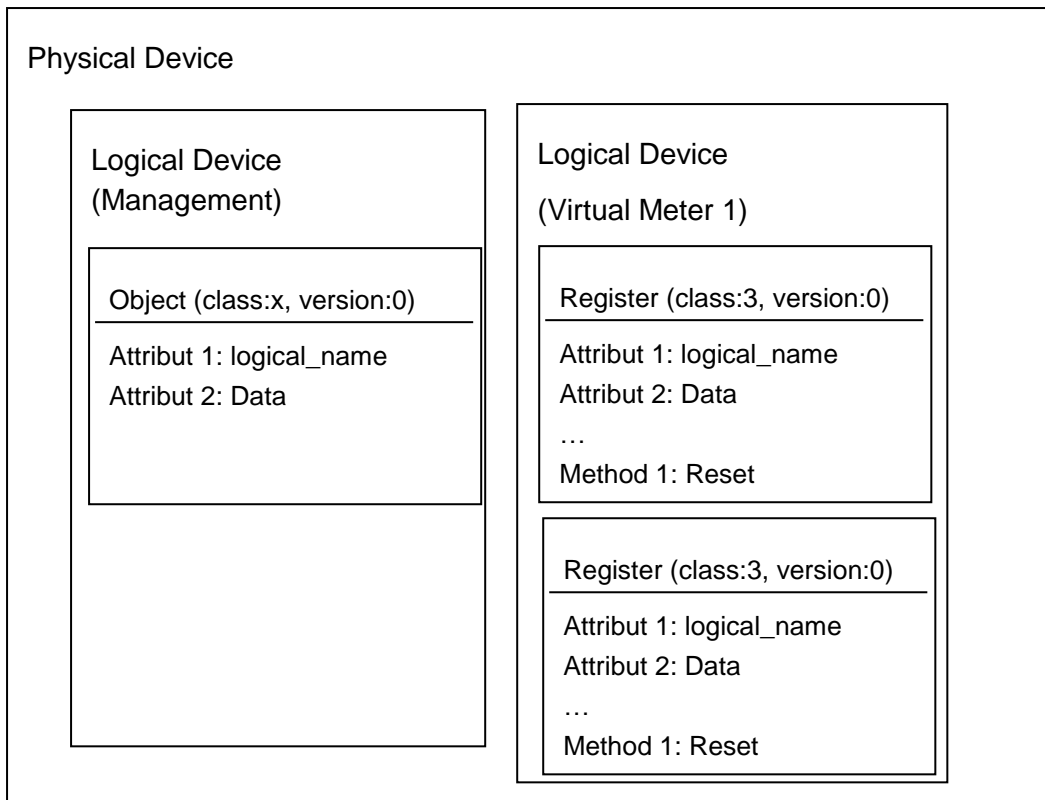
203 • Die HTTP- Session muss nicht nach jeder Transaktion (Request/Response) beendet
204 werden, sondern kann bestehen bleiben (Persistent Connection).

205 5. Datenmodell und API-Struktur

- 206 **5.1. COSEM Physical Device, Logical Device, Object, Attributes, Methods,**
207 Pro Physical COSEM Device MUSS ein Management Logical Device vorhanden sein.
208 Pro Management Logical Device MUSS ein Logical-Device-Name Objekt vorhanden sein.
209 Die Verwendung des Management Logical Device muss noch spezifiziert werden.

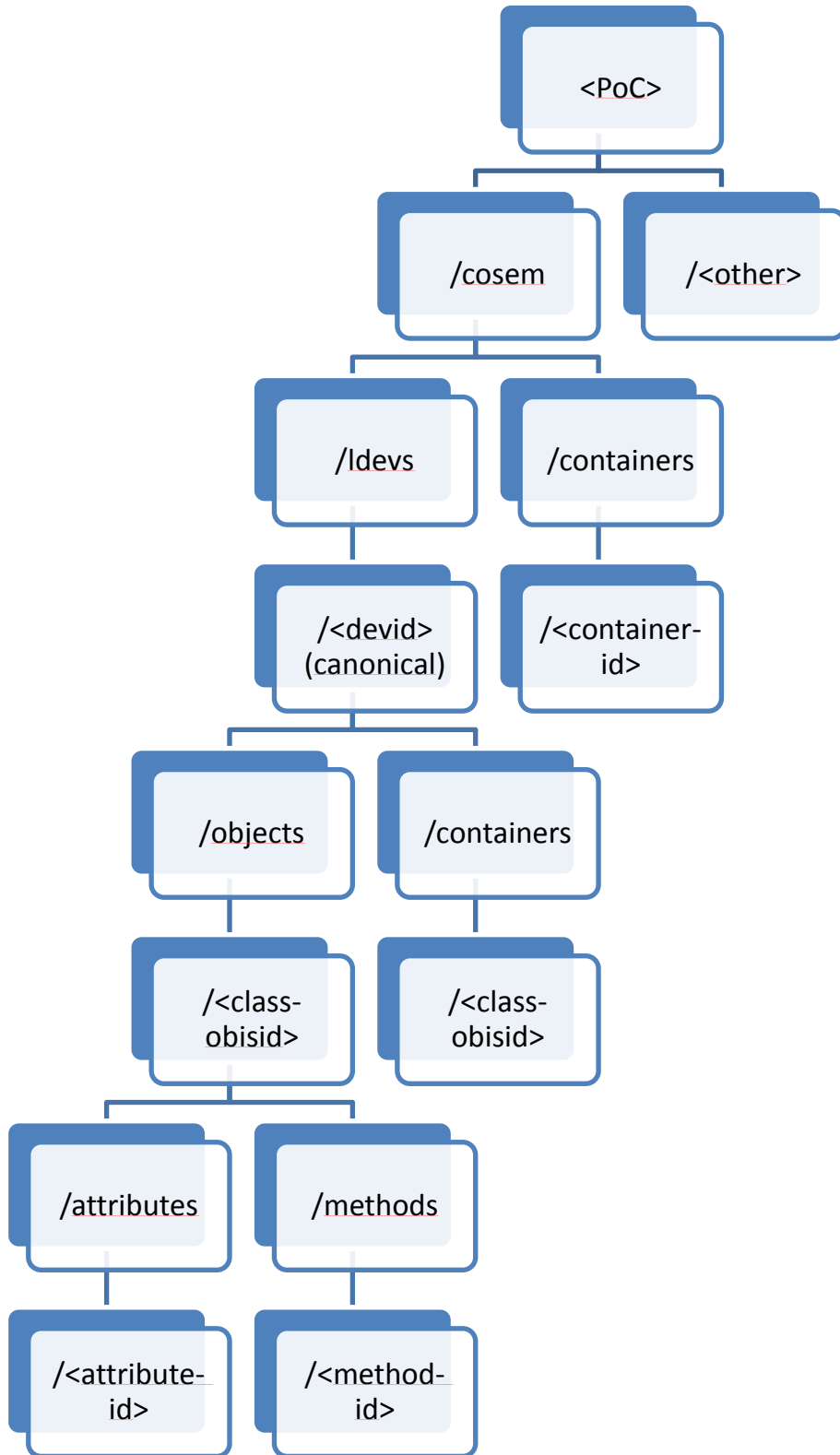
210 **Abbildung 1: Struktur des COSEM Physical Device**

211



212

5.2. URI Resource-Tree eines Physical COSEM Devices



213

214 **Abbildung 2: URI Baum**

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

218 6. Abbildung von Identifikatoren und Bezeichnern

219 6.1. Kanonischer Gerätebezeichner

220 Zur Verwendung verschiedener Gerätenummerierungsschemen in dieser Spezifikation wird
 221 ein kanonischer Gerätebezeichner definiert. Dieser Gerätebezeichner enthält nur die Zeichen
 222 0-9,-,a-z und hat eine Länge von 1-63 Zeichen. Damit ist er geeignet als Hostname
 223 Bestandteil des DNS Systems zu sein.

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

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

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

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

233

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

234 **Tabelle 3: Postfix für Nummerierungsschema**

235 6.2. Hostname

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

238

239 6.3. Logical_DeviceID

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

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

246 Beispiel:

247 Logical_DeviceID: 0x01 MFC1234

248 URI: %01MFC1234

249 XML: 014d464331323334

250

251

252 **6.4. COSEM-Objekt-Bezeichner in der URI**

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

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

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

263 Beispiel:

264 /objects/3-0100010800ff/attributes/3

265

266 **6.5. COSEM Attribut-Index**

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

268

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

269 **Tabelle 4: Attribut Indizes**

270

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

273 **6.6. COSEM Methoden-Index**

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

276

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

279

280 **6.7. Container Bezeichner in der URI**

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

283

284 7. COSEM Zugriff über HTTP

285

286 7.1. HTTP-Request/Response

287

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

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

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

294 Eine Response ohne Request ist nicht möglich.

295

296 Beispiel für einen HTTP-Request:

```
297 GET /path/resource HTTP/1.1<CRLF>
```

```
298 Header1: Tokens<CRLF>
```

```
299 Headern: Tokens<CRLF>
```

```
300 <CRLF>
```

```
301 Optionaler Inhaltsteil ("Body")
```

302

303 Beispiel für einen innerhalb der gleichen Verbindung erzeugten Response:

```
304 HTTP/1.1 Statuscode Statustext<CRLF>
```

```
305 Header1: Tokens<CRLF>
```

```
306 Headern: Tokens<CRLF>
```

```
307 <CRLF>
```

```
308 Optionaler Inhaltsteil ("Body")
```

309

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

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

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

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

322

323 7.1.1. HTTP RESPONSE TIMEOUT

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

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

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

331

332

333 7.1.2. NUTZUNG VON MEHR ALS EINER VERBINDUNG ZWISCHEN KOMMUNIKATIONSPARTNERN

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

339

340 7.2. Zuordnung von Zugriffs-Diensten und HTTP-Verben

341

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

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

347

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
----------------	------	---------------------	-----------------	------	-----

348 **Tabelle 5: Zugriffs-Dienste**

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

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

355 GET, PUT, POST, DELETE operieren genau auf der adressierten Ressource (und evtl.
356 untergeordneten Ressourcen). Bei GET, POST und DELETE muss die Ressource
357 existieren. Bei PUT wird die Ressource angelegt, falls sie nicht existiert.
358

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

359 **Tabelle 6: Idempotenz und Safeness von HTTP-Verben**

360

361

362 **7.3. HTTP-Header**

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

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

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

372 Alle weiteren Header aus [HTTP] KÖNNEN vorhanden sein.

373

374 **7.3.1. REQUEST-HEADER**

375

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.

376 **Tabelle 7: Response Header**

377 M:Muss, B:Bedingt, K:Kann

378 Requests in denen Muss Header Felder fehlen, müssen mit HTTP-Error 400 Bad Request
379 abgelehnt werden.380 Mit x- beginnende Header Felder sind nicht in [HTTP] vorhanden und werden in dieser
381 Spezifikation definiert.

382

383 *7.3.1.1. Content-Type Header*

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

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

386

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

387 **Tabelle 8: Content-Types**

388

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

389 **Tabelle 9: Content-Type Parameter**

390

391 Beispiel:

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

393

394 *7.3.1.2. Content-Encoding Header*

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

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

397

Encoding	Inhalt
deflate	Kompression vor der Encapsulation

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

399

400

401 *7.3.1.3. Content-Length Header*402 Gibt die Länge des folgenden HTTP-Bodies in Bytes an. Dieses Feld MUSS im Request
403 vorhanden sein.

404 Beispiel:

405 Content-Length: 100

406

407 *7.3.1.4. Host Header*408 Enthält den Host-Namen des durch den Request adressierten Servers. Das Feld MUSS im
409 Request vorhanden sein.

410

411 *7.3.1.5. Accept Header*412 Mit diesem HTTP-Header-Feld teilt der Client dem Server mit, welche Content-Typen er im
413 HTTP-Body der Response akzeptiert.414 Sofern eine zusätzliche Spezifikation beschreibt welche Inhaltsdatentypen vom Client
415 akzeptiert werden müssen, KANN der Accept Header entfallen.416 *7.3.1.6. Accept-Encoding Header*417 Mit diesem HTTP-Header-Feld teilt der Client dem Server mit, welche Content-Encodings er
418 im HTTP-Body der Response akzeptiert.419 Sofern eine zusätzliche Spezifikation beschreibt welche Content-Encodings vom Client
420 akzeptiert werden müssen, KANN der Accept-Encoding Header entfallen.

421

422

423 *7.3.1.7. x-CID Header*424 Über die Correlation-ID aus diesem Header-Feld wird der Bezug zwischen Anfrage und
425 Antwort in asynchronen Antworten hergestellt.

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

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

430 Beispiel:

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

432

433 7.3.1.8. Range Header

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

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

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

440 Beispiel:

441 Range: bytes=1024-2047

442

443 7.3.1.9. x-ContactURI Header

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

451 Beispiel:

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

454

455 7.3.1.10. HTTP Kompression

456 Sowohl Server als auch Client MUSS unkomprimierten HTTP Body unterstützen.

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

459

460 7.3.2. RESPONSE-HEADER

461

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.

462 **Tabelle 11: Response-Header**

463 M: MUSS, B:Bedingt, K:KANN

464

465 *7.3.2.1. Content-Type Header*

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

467

468 *7.3.2.2. Content-Encoding Header*

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

470

471 *7.3.2.3. Content-Length Header*

472 Gibt die Länge des folgenden HTTP-Bodies in Bytes an. Dieses Feld MUSS angegeben
 473 werden, wenn ein Content-Body vorhanden ist. Fehlt der Content-Length Header wird ein
 474 leerer Body angenommen. Um „keinen Content“ zu signalisieren wird der HTTP-Status-Code
 475 204 No Content übermittelt.

476

477 *7.3.2.4. Content-Range Header*

478 In der Antwort auf einen Request mit Range-Header-Field liefert der Server den HTTP-
 479 Status-Code 206 Partial Content und gibt im Content-Range-Field an welchen Teil (in
 480 Bytes) einer Ressource der Content-Body enthält.

481 Der Content-Range Header KANN in der Response vorhanden sein.

482

483 Beispiel:

484 Content-Range: bytes 1024-2047/2048

485

486 *7.3.2.5. Retry-after Header*

487

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

491

492 Beispiel:

493

494 Retry-After: 60

495

496

497 **7.4. HTTP Status-Codes**

498

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 wiederholen.
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.

499 **Tabelle 12: HTTP-Statuscodes**

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

502 Die Statuscodes 2xx MUSS als positive Rückmeldung verstanden werden. Nach 202 und
503 206 ist die Kommunikations-Anfrage noch nicht abgeschlossen. Die Positiv-Rückmeldung
504 kann noch nicht an die Anwendung kommuniziert werden bis nicht 200 ok empfangen
505 wurde.

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

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

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

513 Nach dem Empfang eines Fehlerstatuscodes kann der Client

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

520

521 7.5. Synchron/Asynchrone Antwort

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

525

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

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

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

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

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

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

538 Beispiel Head-End-System sendet Request an Gateway:

539

540 POST <GW-PoC>/cosem/.../methods/1 HTTP/1.1

541 x-ContactURI: https://<hostaddr>/<HES-PoC>/cosem/.../methods/2
 542 x-CID: uuid: f50701ab-5a39-4fa6-853f-58aef4297228
 543 *(More Headers)*
 544
 545 *(Content-Body enthält Methoden Input Parameter)*
 546

547 Gateway sendet Response an Head-End-System:
 548 HTTP/1.1 202 Accepted
 549 Content-Length: 0

550

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

554 Gateway → Head-End-System:
 555 POST /<HES-PoC> /cosem/.../methods/2 HTTP/1.1
 556 x-CID: uuid: f50701ab-5a39-4fa6-853f-58aef4297228
 557 *(Content Body enthält Methoden Output-Parameter/Resultat)*
 558

559 Head-End-System → Gateway:
 560
 561 HTTP/1.1 200 OK
 562 Content-length: 0

563

564 7.6. Zugriffsrechte

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

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

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

574 Eine weitergehende Client-Authentifizierung innerhalb von HTTP durch *www-authenticate-*
 575 *Header* ist nach dieser Spezifikation nicht vorgesehen.

576

577

578 7.7. Repräsentation von COSEM Literalen in der HTTP-Zugriffsschicht

579 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	

580 **Tabelle 13: Mapping von COSEM-Literalen in der URI**

581

582 **7.8. Query Parameter**

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

587

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

589 `<path>?attr1=value1{&attrn=valuen}`

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

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

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

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

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

597

598 7.8.1. SELEKTIVER ZUGRIFF AUF INHALTE VON COSEM-ATTRIBUTEN

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

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

605

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

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

607

608 7.8.2. UNIVERSELLE QUERY PARAMETER

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

612

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

613 **Tabelle 15: Universelle Query Parameter**

614

615

616 **sa.retrievecolumns/sa.filtercolumns Descriptor:**617 `{<class>-<logical_name>[,<attribute_index>[,<data_index>]]}`

618 Die Codierung der Literale entspricht Kap. 7.7

619 Die Wertebereiche der Indizes gelten entsprechend [COSEM].

620

621 Beispiel:

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

623 Class 3 – Register

624 1-0:1.8.0- Energy

625 Attribute Index 2 – Value

626 Datenindex 5 – Array Index im Profile-Generic Attribut “buffer”

627

628 7.8.3. BILDUNGSREGEL/SYNTAX FÜR SELECTIVE-ACCESS/QUERY-PARAMETER

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

630 Generische Query-Parameter dieser Spezifikation erhalten einen Präfix q.

631 Die Codierung der Literale entspricht Kap. 7.7

632

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

636

637

638 **7.9. Container-Konzept zum atomaren Zugriff auf aggregierte Ressourcen**

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

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

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

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

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

653

654 **7.10. Listen - Ressourcen**

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

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

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

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

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

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

666

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

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

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

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

675

676 GET <path>/objects HTTP/1.1

677 liefert

678

```
679 <objects count="2" >  
680   <object class-id="3" version="0" id="0100010801ff" >  
681     <attributes>  
682       <attribute id="1">  
683         ..  
684       </attribute>  
685     </attributes>  
686   </object>  
687   <object class-id="3" version="0" id="0100010802ff" >  
688     <attributes>  
689       <attribute id="1">  
690         ..  
691     </attribute>  
692   </attributes>  
693 </object>  
694 </objects>
```

695

696 GET <path>/objects/ HTTP/1.1

697 oder

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

699 liefert

700

```
701 <objects count="2" >  
702   <object class-id="3" version="0" id="0100010801ff" />  
703   <object class-id="3" version="0" id="0100010802ff" />  
704 </objects>  
705
```

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

707 liefert

```
708 <objects count="2" />  
709
```

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

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

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

716

7.11.1. CREATE RESSOURCE

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

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

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

725

726 Beispiel:

727 Client → Server:

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

729

730 {Body}

731 XML-Daten der Objekt-Instanz.

732

733 Server → Client:

734 HTTP/1.1 201 Created

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

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

738

7.11.2. DELETE RESOURCE

740

741 Durch HTTP-DELETE wird eine Ressource gelöscht.

742 Client → Server:

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

744

745 Server → Client:

746 HTTP/1.1 204 No content

747

748 7.12. Fragmentierung von Inhaltsdaten bei Abbruch von Transport-Verbindungen

749

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

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

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

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

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

769

770 7.12.1. ÜBERTRAGEN VON GROSSEN HTTP-BODIES DURCH BLOCKTRANSFER IN HTTP

771

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

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

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

777

778 Beispiel Lesen einer Ressource von 1000 Bytes in zwei Blöcken:

779 Request:

780 GET /largecontent http/1.0

781 Host: <host>

782 Range: bytes 0-511

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

784 Response:

785 HTTP/1.1 206 Partial Content
786 Content-range: bytes 0-511/1000
787 Content-length: 512
788 { Data }
789

790 **Request:**
791 GET /largecontent http/1.0
792 Host: <host>
793 Range: bytes 512-

794

795 **Response:**
796 HTTP/1.1 206 Partial Content
797 Content-range: bytes 512-999/1000
798 Content-length: 488
799 { Data }

800

801 Hinweis: Der Server kann den Zwischenspeicher hier freigeben.

802

803

804 **8. Inhaltsdaten (http Content-Body)**

805 **8.1. Hinweis zum Zugriffslayer**

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

810

811 **8.2. URI Resource-Tree eines Physical COSEM Devices**

812 8.2.1. DER POINT-OF-CONTACT (<POC>)

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

815

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

818 8.2.2. XML INHALTS-CODIERUNG

819 Der XML Inhalt MUSS in UTF-8 codiert werden:

820 Beispiel:

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

822

823 8.2.3. XSD SCHEMA RELATION

824

825

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

827

828 Beispiel:

```
829 <xsd:schema
```

```
830     xmlns:cod="urn:k461-dke-de:cod-1"
```

```
831     attributeFormDefault="unqualified"
```

```
832     elementFormDefault="qualified"
```

```
833     xmlns:cor="urn:k461-dke-de:cor-1"
```

```
834     targetNamespace="urn:k461-dke-de:cor-1"
```

```
835     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

836

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

838

839 8.2.4. COSEM ELEMENT – WURZEL DES COSEM PHYSICAL DEVICE

840

841 Zulässige Operationen auf der Ressource

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


```
844
845 Schema:
846 <xsd:complexType name="cosemType">
847 <xsd:sequence>
848 <xsd:element name="ldevs" type="ldevsType" minOccurs="1" maxOccurs="unbounded" />
849 <xsd:element name="containers" type="containersType" minOccurs="0"
850 maxOccurs="unbounded"/>
851 </xsd:sequence>
852 </xsd:complexType>
853
854 <xsd:element name="cosem" type="cosemType" />
855
```

```
856 Beispiel:
857 <cosem>
858 <ldevs count="1">
859 ...
860 </ldevs>
861 <containers count="1">
862 ...
863 </containers>
864 </cosem>
```

865

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

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

870

871 8.2.5. LVDEVS ELEMENT – LISTE DER LOGICAL DEVICES

872 Zulässige Operationen auf der Ressource

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

875

```
876 Schema:
877 <xsd:complexType name="ldevsType">
878 <xsd:sequence>
879 <xsd:element name="ldev" type="ldevType" minOccurs="1" maxOccurs="unbounded" />
880 </xsd:sequence>
881 <xsd:attribute name="count" type="xsd:unsignedInt" use="optional" />
882 </xsd:complexType>
883
884 <xsd:element name="ldevs" type="ldevsType" />
885
```

```
886 Beispiel:
887 <ldevs count="1">
888 <ldev id="1din0a12345678.sm[.<domain>]" />
889 </ldevs>
```

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

892

893 8.2.6. LDEV ELEMENT – WURZEL DES LOGICAL DEVICE

894 Zulässige Operationen auf der Ressource

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

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

900 DELETE <poc>/cosem/ldevs/<ldevid> – Löscht dynamisch ein Logical Device (ggf. mit
901 Objekten)

902 Hinweis: Das Management Logical Device kann nicht gelöscht werden.

903 Schema:

```
904 <xsd:complexType name="ldevType">
905 <xsd:sequence>
906 <xsd:element name="objects" type="objectsType" minOccurs="1" maxOccurs="unbounded"
907 />
908 <xsd:element name="containers" type="ldevContainersType" minOccurs="0"
909 maxOccurs="unbounded" />
910 </xsd:sequence>
911 <xsd:attribute name="id" type="cod:octet-string" use="required" />
912 </xsd:complexType>
913
914 <xsd:element name="ldev" type="ldevType" />
```

915 Beispiel:

```
916 <ldev id="1din0a12345678.sm[.domain]" >
917 <objects count="1" >
918 ...
919 </objects>
920 </ldev>
```

924

925 8.2.7. OBJECTS ELEMENT – LISTE ALLER OBJEKTE EINES LOGICAL DEVICE

926

927 Zulässige Operationen auf der Ressource

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

930

931 Schema:

```
932 <xsd:complexType name="objectsType">
933 <xsd:sequence>
934 <xsd:element name="object" type="objectType" />
935 </xsd:sequence>
936 <xsd:attribute name="count" type="xsd:unsignedInt" use="optional"/>
937 </xsd:complexType>
938
939 <xsd:element name="objects" type="objectsType" />
```

940

941 Beispiel:

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

```

944 <object id="0100010802ff" class-id="3" version="0" />
945 </objects>
946

```

947 8.2.8. OBJECT ELEMENT –COSEM OBJEKT

948

949 Zulässige Operationen auf der Ressource

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

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

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

957

958

959 Schema:

```

960 <xsd:complexType name="objectType">
961 <xsd:sequence>
962 <xsd:element name="attributes" type="attributesType" minOccurs="1"
963 maxOccurs="unbounded" />
964 <xsd:element name="methods" type="methodsType" minOccurs="0" maxOccurs="unbounded"
965 />
966 </xsd:sequence>
967 <xsd:attribute name="id" type="cod:Cosem-Object-Instance-Id" use="required"/>
968 <xsd:attribute name="class-id" type="cod:Cosem-Class-Id" use="required"/>
969 <xsd:attribute name="version" type="Unsigned8" use="required"/>
970 </xsd:complexType>
971
972 <xsd:element name="object" type="objectType" />
973
974

```

974

975 Beispiel:

```

976 <object class-id="3" version="0" id="0100010800ff">
977 <attributes count="1" >
978 (Data)
979 </attributes>
980 <methods count="1" >
981 </methods>
982 </object>

```

983

984 8.2.9. ATTRIBUTES ELEMENT –LISTE ALLER ATTRIBUTE EINES COSEM OBJEKTES

985

986 Zulässige Operationen auf der Ressource

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

989

990 Schema:

```

991 <xsd:complexType name="attributesType">
992 <xsd:sequence>

```

```

993 <xsd:element name="attribute" type="attributeType" minOccurs="1"
994 maxOccurs="unbounded" />
995 </xsd:sequence>
996 <xsd:attribute name="count" type="xsd:unsignedInt" use="optional"/>
997 </xsd:complexType>
998
999
1000 <xsd:element name="attributes" type="attributesType" />
1001
1002

```

Beispiel:

```

1003
1004
1005 <attributes count="2">
1006 <attribute id="1" >
1007 <octet-string>0100010800ff</octet-string>
1008 </attribute>
1009 <attribute id="2" >
1010 <Integer8>1</Integer8>
1011 </attribute>
1012 </attributes>

```

1013

1014 8.2.10. ATTRIBUTE ELEMENT –COSEM ATTRIBUT

1015

1016 Zulässige Operationen auf der Ressource

1017 GET <poc>/cosem/ldevs/<ldevid>/objects/<objid>/attributes/<attrid> – Liefert den Inhalt
1018 eines Attributes eines Objektes als Response-Body

1019 PUT <poc>/cosem/ldevs/<ldevid>/objects/<objid>/attributes/<attrid> – Überschreibt den
1020 Inhalt eines Attributes mit dem Inhalt des Request-Bodies

1021

1022 Schema:

```

1023
1024 <xsd:complexType name="attributeType">
1025 <xsd:sequence>
1026 <xsd:element name="attribute" type="cod:Data" min_occurs="1"
1027 max_occurs="1" />
1028 </xsd:sequence>
1029 <xsd:attribute name="id" type="cod:Cosem-Attribute-Id-Type"
1030 use="required" />
1031 </xsd:complexType>
1032

```

1033 **Beispiel:**

```

1034 <attribute id="1" >
1035 <octet-string>0100010801ff</octet-string>
1036 </attribute>

```

1037

1038 8.2.11. METHODS ELEMENT –LISTE ALLER METHODEN EINES COSEM OBJEKTES

1039 Zulässige Operationen auf der Ressource

1040 GET <poc>/cosem/ldevs/<ldevid>/objects/<class-objid>/methods – Liefert Liste aller
1041 Methoden eines Objektes als Response-Body

1042

1043 **Schema:**

```

1044 <xsd:complexType name="methodsType">
1045 <xsd:sequence>
1046 <xsd:element name="method" type="methodType" min_occurs="0"
1047 max_occurs="unbounded" />
1048 </xsd:sequence>
1049 <xsd:attribute name="count" type="xsd:unsignedInt" use="optional"/>
1050 </xsd:complexType>
1051

```

1052 **Beispiel:**

```

1053 <methods count="2" ">
1054 <method id="1" />
1055 <method id="2" />
1056 </methods>

```

1057 8.2.12. METHOD ELEMENT –COSEM METHODE

1058

1059 Input-Parameter und Output-Parameter sind vom Typ cod:Data.

1060

1061 Zulässige Operationen auf der Ressource

1062 POST <poc>/cosem/ldevs/<ldevid>/objects/<objid>/methods/<methodid> – Ruft die Methode
 1063 mit den Daten des Request-Bodies auf. Ein Fehler-Code wird im Response-Body übergeben.
 1064 (http-Statuscode 500) Evtl. Output-Parameter werden im Response-Body zurückgeliefert.

1065

1066 **Schema:**

1067

1068 Wird als Response auf GET <...>/methods Listen-Ressource geliefert

1069

```

1070 <xsd:complexType name="methodType">
1071 <xsd:sequence />
1072 <xsd:attribute name="id" type="cod:Cosem-Method-Id-Type"
1073 use="required" />
1074 </xsd:complexType>
1075

```

1076 Wird als Request Body für POST <...>/methods/<id> verwendet:

```

1077 <xsd:element name="methodRequestParameter" type="cod:Data"
1078 min_occurs="0" max_occurs="1" />
1079

```

1080 Wird als Response auf POST <...>/methods/<id> geliefert

1081

```

1082 <xsd:complexType name="methodResponseType">
1083 <xsd:sequence>
1084 <xsd:element name="methodResponseStatus" type="cod:Unsigned8"
1085 min_occurs="0" max_occurs="1" />
1086 <xsd:element name="methodResponseData" type="cod:Data"
1087 min_occurs="0" max_occurs="1" />
1088 </xsd:sequence>
1089 </xsd:complexType>
1090

```

1091

```

1092 <xsd:element name="methodResponse" type="methodResponseType" />

```

1093
 1094 **Beispiel Request-Body:**
 1095 ...
 1096 <methodRequestParameter>
 1097 <Integer8>0</Integer8>
 1098 </methodRequestParameter>
 1099 ...
 1100 **Beispiel Response-Body:**
 1101 ...
 1102 <methodResponse>
 1103 <methodResponseData>
 1104 <Integer8>0</Integer8>
 1105 </methodResponseData>
 1106 </methodRequest>
 1107 ...
 1108 Ein methodResponseStatus=0 wird mit HTTP-Statuscode 200 übermittelt. Ein methodResponseStatus
 1109 >0 wird mit http-Statuscode 500 übermittelt.

1110 8.2.13. CONTAINERS ELEMENT –LISTE ALLER CONTAINER EINES LOGICAL DEVICES

1111 Zulässige Operationen auf der Ressource

1112 GET <poc>/cosem/ldevs/<ldevid>/containers – Liefert Liste aller Container eines
 1113 LogicalDevices als Response-Body

1114

1115 **Schema:**

```
1116 <xsd:complexType name="ldevContainersType">
1117 <xsd:sequence>
1118 <xsd:element name="container" type="ldevContainerType"
1119 min_occurs="0" max_occurs="unbounded" />
1120 </xsd:sequence>
1121 <xsd:attribute name="count" type="xsd:unsignedInt" use="optional"/>
1122 </xsd:complexType>
1123
1124 <xsd:element name="containers" type="ldevContainersType" >
1125
```

1126 **GET/PUT Body Beispiel:**

```
1127 <containers count="1">
1128 <container id="00005e3101ff">
1129 <object class-id="3" version="0" id="0100010801ff">
1130 ...
1131 </object>
1132 <object class-id="3" version="0" id="0100010802ff">
1133 ...
1134 </object>
1135 </container>
1136 </containers>
```

1137 8.2.14. CONTAINER ELEMENT IM LOGICAL DEVICE

1138

1139 Zulässige Operationen auf der Ressource

1140 GET <poc>/cosem/ldevs/<ldevid>/containers/<containerid> – Liefert den Inhalt eines
1141 Containers als Response-Body

1142 PUT <poc>/cosem/ldevs/<ldevid>/containers/<containerid> – Erzeugt oder überschreibt den
1143 Inhalt eines Containers mit dem Inhalt des Request-Bodies
1144

1145 DELETE <poc>/cosem/ldevs/<ldevid>/containers/<containerid> – Entfernt einen Container

```
1146
1147
1148 <xsd:complexType name="ldevContainerType">
1149 <xsd:sequence>
1150 <xsd:element name="container" type="objectsType" />
1151 </xsd:sequence>
1152 <xsd:attribute name="id" type="cod:Cosem-Object-Instance-Id" />
1153 </xsd:complexType>
1154
1155 <xsd:element name="container" type="ldevContainerType" >
1156
```

1157 GET/PUT Body Beispiel:

```
1158 <containers count="1" ">
1159 <container id="00005e3101ff">
1160 <object class-id="3" version="0" id="0100010801ff">
1161 ...
1162 </object>
1163 <object class-id="3" version="0" id="0100010802ff">
1164 ...
1165 </object>
1166 </container>
1167 </containers>
1168
```

1169 8.2.15. CONTAINERS ELEMENT –LISTE ALLER CONTAINER

1170 Zulässige Operationen auf der Ressource

1171 GET <poc>/cosem/containers – Liefert Liste aller Container als Response-Body

1172

1173 Schema:

```
1174 <xsd:complexType name="containersType">
1175 <xsd:sequence>
1176 <xsd:element name="container" type="containerType" minOccurs="0"
1177 maxOccurs="unbounded" />
1178 </xsd:sequence>
1179 <xsd:attribute name="count" type="xsd:unsignedInt" use="optional"/>
1180 </xsd:complexType>
1181
```

1182 8.2.16. CONTAINER ELEMENT

1183

1184 Zulässige Operationen auf der Ressource

1185 GET <poc>/cosem/containers/<containerid> – Liefert den Inhalt eines Containers als
1186 Response-Body

1187 PUT <poc>/cosem/containers/<containerid> – Erzeugt oder überschreibt den Inhalt eines
1188 Containers mit dem Inhalt des Request-Bodies
1189

1190 DELETE <poc>/cosem/containers/<containerid> – Entfernt einen Container

```
1191  
1192  
1193 <xsd:complexType name="containerType">  
1194 <xsd:sequence>  
1195 <xsd:element name="container" type="ldevsType" />  
1196 </xsd:sequence>  
1197 <xsd:attribute name="id" type="hexBinary" />  
1198 </xsd:complexType>
```

1199
1200
1201

1202

1203 8.3. Datentypen für COSEM-Objekte und Attribute

1204 Die Common Datatypes aus IEC62056-6-2:2010 Kap 4.5 werden unterstützt.

```
1205  
1206 <?xml version="1.0" encoding="UTF-8" ?>  
1207  
1208 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:k461-dke-  
1209 de:cod-1 -targetNamespace="urn:k461-dke-de:cod-1" elementFormDefault="qualified">
```

```
1210  
1211 <xsd:simpleType name="Integer8">  
1212 <xsd:restriction base="xsd:byte" />  
1213 </xsd:simpleType>
```

```
1214  
1215 <xsd:simpleType name="Integer16">  
1216 <xsd:restriction base="xsd:short" />  
1217 </xsd:simpleType>
```

```
1218  
1219 <xsd:simpleType name="Integer32">  
1220 <xsd:restriction base="xsd:int" />  
1221 </xsd:simpleType>
```

```
1222  
1223 <xsd:simpleType name="Integer64">  
1224 <xsd:restriction base="xsd:long" />  
1225 </xsd:simpleType>
```

```
1226  
1227 <xsd:simpleType name="Unsigned8">  
1228 <xsd:restriction base="xsd:unsignedByte" />  
1229 </xsd:simpleType>
```

```
1230  
1231 <xsd:simpleType name="Unsigned16">  
1232 <xsd:restriction base="xsd:unsignedShort" />  
1233 </xsd:simpleType>
```

1234


```
1235 <xsd:simpleType name="Unsigned32">
1236 <xsd:restriction base="xsd:unsignedInt" />
1237 </xsd:simpleType>
1238
1239 <xsd:simpleType name="Unsigned64">
1240 <xsd:restriction base="xsd:unsignedLong" />
1241 </xsd:simpleType>
1242
1243 <xsd:simpleType name="Cosem-Class-Id">
1244 <xsd:restriction base="Unsigned16" />
1245 </xsd:simpleType>
1246
1247 <xsd:simpleType name="Cosem-Object-Instance-Id">
1248 <xsd:restriction base="xsd:hexBinary" >
1249 </xsd:simpleType>
1250
1251 <xsd:simpleType name="Cosem-Object-Attribute-Id">
1252 <xsd:restriction base="Integer8" />
1253 </xsd:simpleType>
1254
1255 <xsd:simpleType name="Cosem-Object-Method-Id">
1256 <xsd:restriction base="Integer8" />
1257 </xsd:simpleType>
1258
1259 <xsd:complexType name="NULL" final="#all" />
1260
1261 <xsd:simpleType name="BitString">
1262 <xsd:restriction base="xsd:string">
1263 <xsd:pattern value="[0-1]{0,}" />
1264 </xsd:restriction>
1265 </xsd:simpleType>
1266
1267
1268 <xsd:complexType name="TypeDescription">
1269 <xsd:choice>
1270
1271 <xsd:element name="null-data" type="NULL" />
1272
1273 <xsd:element name="array">
1274 <xsd:complexType>
1275 <xsd:sequence>
1276 <xsd:element name="number-of-elements" type="Unsigned16" />
1277 <xsd:element name="type-description" type="TypeDescription" />
1278 </xsd:sequence>
1279 </xsd:complexType>
1280 </xsd:element>
1281
1282 <xsd:element name="structure">
1283 <xsd:complexType>
1284 <xsd:sequence minOccurs="0" maxOccurs="unbounded">
1285 <xsd:element name="TypeDescription" type="TypeDescription" />
1286 </xsd:sequence>
1287 </xsd:complexType>
1288 </xsd:element>
```

```
1289
1290 <xsd:element name="boolean" type="NULL" />
1291
1292 <xsd:element name="bit-string" type="NULL" />
1293
1294 <xsd:element name="double-long" type="NULL" />
1295
1296 <xsd:element name="double-long-unsigned" type="NULL" />
1297
1298 <xsd:element name="floating-point" type="NULL" />
1299
1300 <xsd:element name="octet-string" type="NULL" />
1301
1302 <xsd:element name="visible-string" type="NULL" />
1303
1304 <xsd:element name="UTF8-string" type="NULL" />
1305
1306 <xsd:element name="bcd" type="NULL" />
1307
1308 <xsd:element name="integer" type="NULL" />
1309
1310 <xsd:element name="long" type="NULL" />
1311
1312 <xsd:element name="unsigned" type="NULL" />
1313
1314 <xsd:element name="long-unsigned" type="NULL" />
1315
1316 <xsd:element name="long64" type="NULL" />
1317
1318 <xsd:element name="long64-unsigned" type="NULL" />
1319
1320 <xsd:element name="enum" type="NULL" />
1321
1322 <xsd:element name="float32" type="NULL" />
1323
1324 <xsd:element name="float64" type="NULL" />
1325
1326 <xsd:element name="date_time" type="NULL" />
1327
1328 <xsd:element name="date" type="#NULL" />
1329
1330 <xsd:element name="time" type="NULL" />
1331
1332 <xsd:element name="dont-care" type="NULL" />
1333
1334 </xsd:choice>
1335 </xsd:complexType>
1336
1337
1338
1339 <xsd:complexType name="SequenceOfData">
1340 <xsd:choice minOccurs="0" maxOccurs="unbounded">
1341
1342 <xsd:element name="null-data" type="NULL" />
1343
```

```
1344 <xsd:element name="array" type="SequenceOfData" />
1345
1346 <xsd:element name="structure" type="SequenceOfData" />
1347
1348 <xsd:element name="boolean" type="xsd:boolean" />
1349
1350 <xsd:element name="bit-string" type="BitString" />
1351
1352 <xsd:element name="double-long" type="Integer32" />
1353
1354 <xsd:element name="double-long-unsigned" type="Unsigned32" />
1355
1356 <xsd:element name="floating-point">
1357 <xsd:simpleType>
1358 <xsd:restriction base="xsd:hexBinary">
1359 <xsd:length value="4" />
1360 </xsd:restriction>
1361 </xsd:simpleType>
1362 </xsd:element>
1363
1364 <xsd:element name="octet-string" type="xsd:hexBinary" />
1365
1366 <xsd:element name="visible-string" type="xsd:string" />
1367
1368 <xsd:element name="UTF8-string" type="xsd:string" />
1369
1370 <xsd:element name="bcd" type="Integer8" />
1371
1372 <xsd:element name="integer" type="Integer8" />
1373
1374 <xsd:element name="long" type="Integer16" />
1375
1376 <xsd:element name="unsigned" type="Unsigned8" />
1377
1378 <xsd:element name="long-unsigned" type="Unsigned16" />
1379
1380 <xsd:element name="compact-array">
1381 <xsd:complexType>
1382 <xsd:sequence>
1383 <xsd:element name="contents-description" type="TypeDescription" />
1384 <xsd:element name="array-contents" type="xsd:hexBinary" />
1385 </xsd:sequence>
1386 </xsd:complexType>
1387 </xsd:element>
1388
1389 <xsd:element name="long64" type="Integer64" />
1390
1391 <xsd:element name="long64-unsigned" type="Unsigned64" />
1392
1393 <xsd:element name="enum" type="Unsigned8" />
1394
1395 <xsd:element name="float32">
1396 <xsd:simpleType>
1397 <xsd:restriction base="xsd:hexBinary">
```

```
1399 <xsd:length value="4" />
1400 </xsd:restriction>
1401 </xsd:simpleType>
1402 </xsd:element>
1403
1404 <xsd:element name="float64">
1405 <xsd:simpleType>
1406 <xsd:restriction base="xsd:hexBinary">
1407 <xsd:length value="8" />
1408 </xsd:restriction>
1409 </xsd:simpleType>
1410 </xsd:element>
1411
1412 <xsd:element name="date_time">
1413 <xsd:simpleType>
1414 <xsd:restriction base="xsd:hexBinary">
1415 <xsd:length value="12" />
1416 </xsd:restriction>
1417 </xsd:simpleType>
1418 </xsd:element>
1419
1420 <xsd:element name="date">
1421 <xsd:simpleType>
1422 <xsd:restriction base="xsd:hexBinary">
1423 <xsd:length value="5" />
1424 </xsd:restriction>
1425 </xsd:simpleType>
1426 </xsd:element>
1427
1428 <xsd:element name="time">
1429 <xsd:simpleType>
1430 <xsd:restriction base="xsd:hexBinary">
1431 <xsd:length value="4" />
1432 </xsd:restriction>
1433 </xsd:simpleType>
1434 </xsd:element>
1435
1436 <xsd:element name="dont-care" type="NULL" />
1437
1438 </xsd:choice>
1439 </xsd:complexType>
1440
1441
1442
1443 <xsd:complexType name="Data">
1444 <xsd:choice>
1445
1446 <xsd:element name="null-data" type="NULL" />
1447
1448 <xsd:element name="array" type="SequenceOfData" />
1449
1450 <xsd:element name="structure" type="SequenceOfData" />
1451
1452 <xsd:element name="boolean" type="xsd:boolean" />
1453
```

```
1454 <xsd:element name="bit-string" type="BitString" />
1455
1456 <xsd:element name="double-long" type="Integer32" />
1457
1458 <xsd:element name="double-long-unsigned" type="Unsigned32" />
1459
1460 <xsd:element name="floating-point">
1461 <xsd:simpleType>
1462 <xsd:restriction base="xsd:hexBinary">
1463 <xsd:length value="4" />
1464 </xsd:restriction>
1465 </xsd:simpleType>
1466 </xsd:element>
1467
1468 <xsd:element name="octet-string" type="xsd:hexBinary" />
1469
1470 <xsd:element name="visible-string" type="xsd:string" />
1471
1472 <xsd:element name="UTF8-string" type="xsd:string" />
1473
1474 <xsd:element name="bcd" type="Integer8" />
1475
1476 <xsd:element name="integer" type="Integer8" />
1477
1478 <xsd:element name="long" type="Integer16" />
1479
1480 <xsd:element name="unsigned" type="Unsigned8" />
1481
1482 <xsd:element name="long-unsigned" type="Unsigned16" />
1483
1484 <xsd:element name="compact-array">
1485 <xsd:complexType>
1486 <xsd:sequence>
1487 <xsd:element name="contents-description" type="TypeDescription" />
1488 <xsd:element name="array-contents" type="xsd:hexBinary" />
1489 </xsd:sequence>
1490 </xsd:complexType>
1491 </xsd:element>
1492
1493 <xsd:element name="long64" type="Integer64" />
1494
1495 <xsd:element name="long64-unsigned" type="Unsigned64" />
1496
1497 <xsd:element name="enum" type="Unsigned8" />
1498
1499 <xsd:element name="float32">
1500 <xsd:simpleType>
1501 <xsd:restriction base="xsd:hexBinary">
1502 <xsd:length value="4" />
1503 </xsd:restriction>
1504 </xsd:simpleType>
1505 </xsd:element>
1506
1507 <xsd:element name="float64">
1508 <xsd:simpleType>
```

```
1509 <xsd:restriction base="xsd:hexBinary">
1510 <xsd:length value="8" />
1511 </xsd:restriction>
1512 </xsd:simpleType>
1513 </xsd:element>
1514
1515 <xsd:element name="date_time">
1516 <xsd:simpleType>
1517 <xsd:restriction base="xsd:hexBinary">
1518 <xsd:length value="12" />
1519 </xsd:restriction>
1520 </xsd:simpleType>
1521 </xsd:element>
1522
1523 <xsd:element name="date">
1524 <xsd:simpleType>
1525 <xsd:restriction base="xsd:hexBinary">
1526 <xsd:length value="5" />
1527 </xsd:restriction>
1528 </xsd:simpleType>
1529 </xsd:element>
1530
1531 <xsd:element name="time">
1532 <xsd:simpleType>
1533 <xsd:restriction base="xsd:hexBinary">
1534 <xsd:length value="4" />
1535 </xsd:restriction>
1536 </xsd:simpleType>
1537 </xsd:element>
1538
1539 <xsd:element name="dont-care" type="NULL" />
1540 </xsd:choice>
1541 </xsd:complexType>
1542
1543 </xsd:schema>
1544
1545
```