

1



2

3

4



5

6

7

8

Smart Energy Profile 2 Application Protocol Standard

9

10

11

12

ZigBee Public Document 13-0200-00

April 2013

Sponsored by: ZigBee Alliance

Accepted by This document has been accepted for release by the ZigBee Alliance Board of Directors.

Abstract This document describes Smart Energy Profile 2, an IP-based standard for energy management and communications.

Keywords Smart Energy Profile 2, SEP 2, Demand Response and Load Control, Pricing Communication, Energy Usage Information, Metering, Prepayment, Plugin Electric Vehicles, Distributed Energy Resources, RESTful

13

14

15

16

April 2013

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

This page is intentionally blank.

38 Notice of use and disclosure

39 Copyright 2011-2013 © ZigBee Alliance, Inc. and HomePlug Powerline Alliance, Inc. All rights Reserved. This
40 information within this document is the property of the ZigBee Alliance and the HomePlug Powerline Alliance and
41 its use and disclosure are restricted.

42 Elements of ZigBee Alliance and HomePlug Powerline Alliance specifications may be subject to third party
43 intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party
44 MAY or MAY NOT be a member of ZigBee and / or HomePlug). ZigBee and HomePlug are not responsible and
45 SHALL NOT be held responsible in any manner for identifying or failing to identify any or all such third party
46 intellectual property rights.

47 This document and the information contained herein are provided on an "AS IS" basis and ZigBee and HomePlug
48 DISCLAIM ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY
49 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF
50 THIRD PARTIES (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS
51 INCLUDING PATENT, COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF
52 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT. IN
53 NO EVENT WILL ZIGBEE OR HOMEPLUG BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF
54 BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT,
55 INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF
56 ANY KIND, IN CONTRACT OR IN TORT, IN CONNECTION WITH THIS DOCUMENT OR THE
57 INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR
58 DAMAGE. All Company, brand and product names may be trademarks that are the sole property of their respective
59 owners.

60 The above notice and this paragraph MUST be included on all copies of this document that are made.

61 ZigBee Alliance, Inc. HomePlug Powerline Alliance, Inc.
62 2400 Camino Ramon, Suite 375 5200 SW Macadam Ave., Suite 470
63 San Ramon, CA 94583 Portland, OR 97239

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

This page is intentionally blank.

82 Foreword

83 The empowerment of consumers to manage their usage and generation of energy is a critical feature of
84 the Smart Grid and is a basis of innovation for new products and services in energy management. To
85 enable this capability, information flow between devices such as meters, smart appliances, plug-in electric
86 vehicles, energy management systems, and distributed energy resources (including renewable energy and
87 storage elements) must occur in an open, standardized, secure, and interoperable fashion. The following
88 specification is intended to fulfill those needs.

89 The development of this document has been driven by, and seeks to address the requirements of, many
90 activities across the globe. Of note are the efforts within the United States by the National Institute of
91 Standards and Technology (NIST) and the Smart Grid Interoperability Panel (SGIP) (in particular,
92 Priority Action Plans 3, 9, 10, 11, and 18, with influence from many of the others) in fulfillment of the
93 EISA 2007 legislation, the European Mandate on Smart Metering (M / 441) (in particular, efforts within
94 CEN / CENELEC and ETSI, and the Smart Meter Working Group), as well as similar efforts in Australia,
95 the United Kingdom, Japan, and China, and electric vehicle standardization efforts (in particular, ISO /
96 IEC JWG automotive EV standards and SAE EV standards), to name only a few.

97 Readers should note that this document was readied for balloting under the policies set forth in the
98 ZigBee Alliance. Previous revisions of this document have gone through review both within the ZigBee
99 Alliance as well as made available for public comment. All previously received comments have been
100 considered for this specification. This specification is intended to meet the requirements set forth in the
101 previously published Technical Requirements Document (TRD).

102 This document is also intended to enable communications that are link-layer agnostic and run over the
103 Internet Protocol. Careful consideration was given to premises networks with various architectures,
104 numbers of devices, and constraints while maintaining flexibility, extensibility, and security.

105

Table of Contents

106	Smart Energy Profile 2 Application Protocol Standard	1
107	1 Introduction	17
109	1.1 Purpose.....	17
110	1.2 Scope	17
111	1.3 Context Overview	17
112	1.4 Document Organization	17
113	1.5 Requirement Language.....	18
114	1.6 Typography Conventions Used.....	18
115	1.7 Design Principles	18
116	2 Acknowledgements.....	19
117	3 Document Revision History	20
118	4 References	21
119	4.1 Smart Energy 2.0 Documents.....	21
120	4.2 IEC Documents	21
121	4.3 IETF Documents.....	21
122	4.4 Other References.....	22
123	5 Definitions, Acronyms and Abbreviations	24
124	5.1 Acronyms and Abbreviations.....	24
125	5.2 Definitions	25
126	6 Design Pattern.....	28
127	6.1 Protocol Flexibility.....	28
128	6.2 General Rules / Best Practices.....	28
129	6.3 WADL.....	29
130	6.4 Schema.....	29
131	6.5 Uniform Resource Identifiers.....	29
132	6.6 List Resources	30
133	6.6.1 Query String Parameters	30
134	6.7 Resource Design Rules.....	33
135	7 Application Support	35
136	7.1 Overview	35
137	7.2 Use of TCP	35
138	7.3 URI Encoding	35
139	7.4 HTTP Headers	35
140	7.4.1 HTTP Header Field Recommended Usage	36
141	7.5 HTTP Response Codes.....	38
142	7.5.1 Common Responses	38
143	7.5.2 Minimal Understanding	40
144	7.6 Application Payload Syntax.....	41
145	7.6.1 XML Encoding	41
146	7.6.2 EXI Encoding.....	41
147	7.7 Content Negotiation	41
148	7.7.1 Schema Version Negotiation	41
149	8 Security	43
150	8.1 Introduction.....	43
151	8.2 Security Attributes.....	43
152	8.2.1 Local Registration Attributes	43
153	8.2.2 Access Control List (ACL) Attributes.....	44
154	8.3 Device Credentials	48

155	8.3.1	Certificate Fingerprint.....	49
156	8.3.2	Short-form Device Identifier (SFDI).....	49
157	8.3.3	Long-form Device Identifier (LFDI).....	49
158	8.3.4	6-digit PIN code.....	49
159	8.3.5	Registration Code	50
160	8.4	<i>Resource Access Authentication and Authorization context</i>	50
161	8.5	<i>Resource Access Authentication</i>	51
162	8.6	<i>Resource Access Authorization</i>	52
163	8.7	<i>Cipher Suites</i>	52
164	8.8	<i>Default Security Policy</i>	52
165	8.9	<i>Registration</i>	54
166	8.9.1	EndDeviceList	55
167	8.10	<i>Security LogEvents</i>	56
168	8.11	<i>Certificate Management</i>	56
169	8.11.1	Introduction.....	56
170	8.11.2	Certificate Usage – Authentication vs. Authorization	58
171	8.11.3	Manufacturing PKI.....	59
172	8.11.4	General Certificate Format.....	60
173	8.11.5	General Restrictions and Conditions	61
174	8.11.6	Extensions.....	61
175	8.11.7	Additional ASN1 Definitions	62
176	8.11.8	Certificate Profiles	63
177	8.11.9	Device Requirements	67
178	8.11.10	Certificate Verification.....	68
179	8.11.11	Certificate Related Labeling Requirements	68
180	9	Discovery	69
181	9.1	<i>Service Instance</i>	69
182	9.2	<i>Service Name</i>	70
183	9.3	<i>TXT Record</i>	70
184	9.4	<i>Subtype Queries</i>	71
185	9.5	<i>Discovery Procedure</i>	73
186	10	Support Resources	74
187	10.1	<i>Resource Section Outlines</i>	74
188	10.1.1	Overview	74
189	10.1.2	List Ordering	74
190	10.1.3	Application Guidelines / Behavior	74
191	10.1.4	LogEvents	75
192	10.2	<i>Device Capabilities Function Set</i>	75
193	10.2.1	Overview	75
194	10.2.2	List Ordering	75
195	10.2.3	Application Guidelines / Behavior	75
196	10.2.4	LogEvents	75
197	10.3	<i>Self Device Resource</i>	75
198	10.3.1	Overview	75
199	10.3.2	List Ordering	75
200	10.3.3	Application Guidelines / Behavior	75
201	10.3.4	LogEvents	76
202	10.4	<i>End Device Resource</i>	76
203	10.4.1	Overview	76

204	10.4.2	List Ordering	76
205	10.4.3	Application Guidelines / Behavior	76
206	10.4.4	LogEvents	77
207	<i>10.5 Function Set Assignments</i>	77
208	10.5.1	Overview	77
209	10.5.2	List Ordering	78
210	10.5.3	Application Guidelines / Behavior	78
211	10.5.4	LogEvents	78
212	<i>10.6 Subscription / Notification Mechanism</i>	78
213	10.6.1	Overview	78
214	10.6.2	List Ordering	79
215	10.6.3	Application Guidelines / Behavior	79
216	10.6.4	LogEvents	81
217	<i>10.7 Response</i>	81
218	10.7.1	Overview	81
219	10.7.2	List Ordering	81
220	10.7.3	Application Guidelines / Behavior	81
221	10.7.4	LogEvents	85
222	11 Common Resources	86
223	<i>11.1 Time Function Set</i>	86
224	11.1.1	Overview	86
225	11.1.2	List Ordering	86
226	11.1.3	Application Guidelines / Behavior	86
227	11.1.4	LogEvents	87
228	<i>11.2 DeviceInformation Function Set</i>	87
229	11.2.1	Overview	87
230	11.2.2	List Ordering	87
231	11.2.3	Application Guidelines / Behavior	87
232	11.2.4	LogEvents	87
233	<i>11.3 Power Status</i>	88
234	11.3.1	Overview	88
235	11.3.2	List Ordering	88
236	11.3.3	Application Guidelines / Behavior	88
237	11.3.4	LogEvents	88
238	<i>11.4 Network Status</i>	88
239	11.4.1	Overview	88
240	11.4.2	List Ordering	88
241	11.4.3	Application Guidelines / Behavior	89
242	11.4.4	LogEvents	89
243	<i>11.5 LogEvent List</i>	89
244	11.5.1	Overview	89
245	11.5.2	List Ordering	89
246	11.5.3	Application Guidelines / Behavior	90
247	<i>11.6 Configuration Resource</i>	90
248	11.6.1	Overview	90
249	11.6.2	List Ordering	90
250	11.6.3	Application Guidelines / Behavior	90
251	11.6.4	LogEvents	91
252	<i>11.7 File Download Function Set</i>	91

253	11.7.1	File List Resource	91
254	11.7.2	File Status Resource	94
255	12	Smart Energy Resources	96
256	<i>12.1</i>	<i>Common Functionality</i>	<i>96</i>
257	12.1.1	Overview	96
258	12.1.2	Active List Elements.....	96
259	12.1.3	Event Rules and Guidelines	96
260	12.1.4	Randomization	99
261	12.1.5	Multi-Server.....	101
262	<i>12.2</i>	<i>Demand Response and Load Control</i>	<i>102</i>
263	12.2.1	Overview	102
264	12.2.2	List Ordering	102
265	12.2.3	Application Guidelines / Behavior	103
266	12.2.4	LogEvents	105
267	<i>12.3</i>	<i>Metering Function Set</i>	<i>105</i>
268	12.3.1	Overview	105
269	12.3.2	List Ordering	105
270	12.3.3	Application Guidelines / Behavior	105
271	12.3.4	LogEvents	109
272	<i>12.4</i>	<i>Pricing Function Set</i>	<i>111</i>
273	12.4.1	Overview	111
274	12.4.2	List Ordering	111
275	12.4.3	Application Guidelines / Behavior	111
276	12.4.4	LogEvents	115
277	<i>12.5</i>	<i>Messaging Function Set</i>	<i>115</i>
278	12.5.1	Overview	115
279	12.5.2	List Ordering	115
280	12.5.3	Application Guidelines / Behavior	116
281	12.5.4	LogEvents	116
282	<i>12.6</i>	<i>Billing Function Set</i>	<i>116</i>
283	12.6.1	Overview	116
284	12.6.2	List Ordering	117
285	12.6.3	Application Guidelines / Behavior	117
286	12.6.4	LogEvents	119
287	<i>12.7</i>	<i>Prepayment Function Set</i>	<i>119</i>
288	12.7.1	Overview	119
289	12.7.2	List Ordering	119
290	12.7.3	Application Guidelines / Behavior	119
291	12.7.4	LogEvents	121
292	<i>12.8</i>	<i>Energy Flow Reservation Function Set</i>	<i>121</i>
293	12.8.1	Overview	121
294	12.8.2	List Ordering	121
295	12.8.3	Application Guidelines / Behavior	121
296	12.8.4	LogEvents	122
297	<i>12.9</i>	<i>Distributed Energy Resources Function Set</i>	<i>122</i>
298	12.9.1	Overview	122
299	12.9.2	Terminology and Conventions.....	123
300	12.9.3	List Ordering	124
301	12.9.4	Application Guidelines / Behavior	124

302	12.9.5	DER Client Device Requirements.....	127
303	12.9.6	LogEvents	129
304	12.10	<i>Metering Mirror</i>	129
305	12.10.1	Overview	129
306	12.10.2	List Ordering	129
307	12.10.3	Application Guidelines / Behavior	129
308	12.10.4	LogEvents	131
309	13	Manufacturer – Specific Proprietary Extensions	132
310	13.1	Overview	132
311	13.2	xmDNS/DNS-SD.....	132
312	13.3	URLs	132
313	13.4	Resources	132
314	13.5	DeviceCapabilities Resource.....	133
315	14	Appendix A - Web-Application Description Language (INFORMATIVE)	134
316	14.1	<i>Support Resources Section</i>	134
317	14.1.1	Device Capability Function Set	134
318	14.1.2	Self Device Resource Function Set	134
319	14.1.3	End Device Resource Function Set	134
320	14.1.4	Function Set Assignments Function Set	135
321	14.1.5	Subscription/Notification Mechanism Function Set.....	135
322	14.1.6	Response Function Set	136
323	14.2	<i>Common Resources Section</i>	137
324	14.2.1	Time Function Set.....	137
325	14.2.2	Device Information Function Set.....	137
326	14.2.3	Power Status Function Set	138
327	14.2.4	Network Status Function Set.....	138
328	14.2.5	Log/Event Log Function Set	140
329	14.3	<i>Smart Energy Resources Section</i>	140
330	14.3.1	Configuration Resource Function Set	140
331	14.3.2	Software Download Function Set.....	141
332	14.3.3	Demand Response and Load Control Function Set	141
333	14.3.4	Metering Function Set.....	142
334	14.3.5	Pricing Function Set.....	144
335	14.3.6	Messaging Function Set.....	145
336	14.3.7	Billing Function Set	146
337	14.3.8	Prepayment Function Set	149
338	14.3.9	Flow Reservation Function Set	151
339	14.3.10	Distributed Energy Resources Function Set	151
340	15	Appendix B – SEP 2 Model (INFORMATIVE)	155
341	15.1	<i>SEP 2 Package</i>	155
342	15.1.1	DeviceCapability Package	155
343	15.1.2	Common Package	156
344	15.1.3	EndDevice Package.....	175
345	15.1.4	FunctionSetAssignments Package	178
346	15.1.5	Pub-Sub Package	179
347	15.1.6	Response Package	181
348	15.1.7	Time Package.....	183
349	15.1.8	DeviceInformation Package.....	184
350	15.1.9	PowerStatus Package	187

351	15.1.10	NetworkStatus Package.....	189
352	15.1.11	LogEvents Package	194
353	15.1.12	Configuration Package.....	196
354	15.1.13	SoftwareDownload Package.....	198
355	15.1.14	DRLC Package	201
356	15.1.15	Metering Package.....	205
357	15.1.16	Pricing Package.....	212
358	15.1.17	Messaging Package.....	216
359	15.1.18	Billing Package	218
360	15.1.19	Prepayment Package	223
361	15.1.20	FlowReservation Package	228
362	15.1.21	DER Package	230
363	15.1.22	Links Package.....	246
364	16	Appendix C – Examples and Guidelines [INFORMATIVE]	252
365	16.1	<i>Registration: Remote</i>	252
366	16.2	<i>Registration: Local.....</i>	255
367	16.3	<i>Discovery: Function Set Assignment.....</i>	258
368	16.4	<i>Discovery: Without Function Set Assignment.....</i>	260
369	16.5	<i>Discovery: Undirected Without Function Set Assignment</i>	262
370	16.6	<i>Subscription / Notification</i>	263
371	16.7	<i>Demand Response: General</i>	266
372	16.8	<i>Demand Response: Cancel</i>	270
373	16.9	<i>Distributed Energy Resource: General.....</i>	272
374	16.10	<i>Metering: Reading</i>	276
375	16.11	<i>Metering: Interval</i>	281
376	16.12	<i>Metering: Instantaneous</i>	287
377	16.13	<i>Metering: Mirroring</i>	290
378	16.14	<i>Pricing: Time of Use</i>	303
379	16.15	<i>Billing: Billing Period</i>	309
380	16.16	<i>Billing: Historical</i>	312
381	16.17	<i>Billing: Projection</i>	315
382	16.18	<i>File Loading</i>	317
383	16.19	<i>Flow Reservation: General</i>	320
384	16.20	<i>Flow Reservation: Cancel</i>	324
385	16.21	<i>Event Randomization</i>	328
386	16.21.1	<i>Simple Event.....</i>	328
387	16.21.2	<i>Event with Positive Randomized Duration</i>	329
388	16.21.3	<i>Event with Positive Randomized Start</i>	329
389	16.21.4	<i>Event with Negative Randomized Start</i>	330
390	16.21.5	<i>Event with Positive Randomization and Finishing in one Hour</i>	331
391	16.21.6	<i>Event with Negative Randomized Start and at Least One Hour Duration</i>	331
392	16.21.7	<i>Event with Randomized Start and Long Ramp Up.....</i>	332
393	16.21.8	<i>Event with Positive Randomized Start and Fixed End Time</i>	333
394	17	Appendix D – Guidelines [INFORMATIVE]	334
395	17.1	<i>Pricing Implementation Guidelines</i>	334
396	17.1.1	<i>Implementing Common Tariff Designs</i>	334
397	17.1.2	<i>Flat Rate Design</i>	334
398	17.1.3	<i>Time-of-Use Rate Design</i>	335
399	17.1.4	<i>Consumption-based Block Rate Design</i>	336

400	17.1.5	Combined Time-of-Use and Consumption-based Block Rate Design.....	336
401	17.1.6	Coordinated and Uncoordinated Pricing and Metering Servers	338
402	17.2	<i>PEV Implementation Guidelines (subject to work with SAE and ISO / IEC)</i>	339
403			

404 List of Tables

405	Table 7-1 HTTP Headers	36
406	Table 8-1 Local Registration Attributes	43
407	Table 8-2 Local Registration Descriptor Entry	43
408	Table 8-3 ACL Attributes	44
409	Table 8-4 AccessDescriptor Entry	44
410	Table 8-5 SpecificIDDescriptor Entry	45
411	Table 8-6 Example ACL for EndDeviceList.....	47
412	Table 8-7 Example ACL entry for EndDevice at point of registration.....	47
413	Table 8-8 Example local registration entry for registering device.....	48
414	Table 8-9 Example ACL entry for EndDevice at point of client access.....	48
415	Table 8-10 Example SpecificIDDescriptor for EndDevice at point of client access.....	48
416	Table 8-11 Attribute values for Default Security Policy.....	53
417	Table 8-12 Security LogEvents.....	56
418	Table 8-13 TLS Authentication Matrix.....	58
419	Table 8-14 SEP 2 Authorization Matrix.....	58
420	Table 9-1 TXT record parameters	70
421	Table 9-2 Service subtype strings and their corresponding SEP 2 function sets.	72
422	Table 10-1 Function Set List Ordering	74
423	Table 10-2 Example LogEvents.	75
424	Table 10-3 Self Device LogEvents.	76
425	Table 10-4 End Device List Ordering.....	76
426	Table 10-5 End Device LogEvents.	77
427	Table 10-6 Function Set Assignments List Ordering.	78
428	Table 10-7 Subscription / Notification List Ordering.	79
429	Table 10-8 Subscription / Notification LogEvents.....	81
430	Table 10-9 Response List Ordering.	81
431	Table 10-10 Response Types by Function Set.....	82
432	Table 10-11 Response LogEvents.	85
433	Table 11-1 Time LogEvents.	87
434	Table 11-2 Power Status LogEvents.....	88
435	Table 11-3 Network Status List Ordering.....	88
436	Table 11-4 Network Status LogEvents	89
437	Table 11-5 LogEvent List Ordering.....	89
438	Table 11-6 General LogEvents.	90
439	Table 11-7 Configuration LogEvents.	91
440	Table 11-8 File Download List Ordering.....	91
441	Table 11-9 File Download LogEvents.	94
442	Table 12-1 Demand Response and Load Control List Ordering.	102
443	Table 12-2 Metering List Ordering.....	105
444	Table 12-3 Reading Types.	108
445	Table 12-4 Metering LogEvents.	109
446	Table 12-5 Pricing List Ordering.....	111
447	Table 12-6 Pricing LogEvents.	115
448	Table 12-7 Messaging List Ordering.	115
449	Table 12-8 Billing List Ordering.....	117
450	Table 12-9 Prepayment List Ordering.....	119
451	Table 12-10 Prepayment LogEvents.	121
452	Table 12-11 FlowReservation List Ordering.....	121

453	Table 12-12 Flow Reservation LogEvents.	122
454	Table 12-13 Distributed Energy Resources List Ordering.	124
455	Table 12-14 Modes and Attributes for Generator Type DERs.	128
456	Table 12-15 Modes and Attributes for Storage Type DERs.	129
457	Table 12-16 Metering Mirror List Ordering.	129
458	Table 12-17 Metering Mirror LogEvents.....	131
459	Table 16-1 POX Example: Registration Remote.....	252
460	Table 16-2 EXI Example: Registration Remote.	254
461	Table 16-3 POX Example: Registration Local.	256
462	Table 16-4 POX Example: Discovery Function Set Assignment	258
463	Table 16-5 POX Example: Discovery Without Function Set Assignment.	260
464	Table 16-6 POX Example: Discovery Undirected Without Function Set Assignment.	262
465	Table 16-7 POX Example: Subscription / Notification.	264
466	Table 16-8 EXI Example: Subscription / Notification.	265
467	Table 16-9 POX Example: Demand Response – General.	266
468	Table 16-10 POX Example: Demand Response - Cancel.	270
469	Table 16-11 POX Example: Distributed Energy Resource - General.	273
470	Table 16-12 POX Example: Meter Reading.	277
471	Table 16-13 POX Example: Metering Interval.....	282
472	Table 16-14 POX Example: Metering Instantaneous.	287
473	Table 16-15 POX Example: Meter Mirroring.	291
474	Table 16-16 POX Example: Pricing TOU.	304
475	Table 16-17 POX Example: Billing Period.....	309
476	Table 16-18 POX Example: Billing Historical.	312
477	Table 16-19 POX Example: Billing Projection.....	315
478	Table 16-20 File Load: Flow Description	318
479	Table 16-21 POX Example: Flow Reservation - General.	321
480	Table 16-22 POX Example: Flow Reservation - Cancel	325
481	Table 16-23 Ramp Value.	328
482	Table 16-24 Event Ramp Time.	329
483	Table 16-25 Event Start Time.	330
484	Table 16-26 Event Start Time.	330
485	Table 16-27 Event Start Time.	331
486	Table 16-28 Event Start Time.	331
487	Table 16-29 Event Start Time.	332
488	Table 17-1 TOU Rate.....	335
489	Table 17-2 Consumption-based Rates.	336
490	Table 17-3 Tariff Structure.....	336
491		

492 List of Figures

493	Figure 8-1: Example Device Authentication Procedure Using HTTPS Port 443	51
494	Figure 8-2: Device authentication with registration procedure examples using HTTPS	55
495	Figure 8-3: SEP 2 including ZigBee IP deployment.....	57
496	Figure 12-1: Active and Reactive Power Flow Directions as Measured at the DER.....	124
497	Figure 12-2: Example Volt-VAr Curve and Hysteresis	126
498	Figure 12-3: Example Low and High - Voltage Ride Through Curves.....	126
499	Figure 13-1: Allowed and Disallowed Extension.....	132
500	Figure 15-1: Version Information	155
501	Figure 15-2: DeviceCapability	155
502	Figure 15-3: Identification	156
503	Figure 15-4: Events	159
504	Figure 15-5: Programs	159
505	Figure 15-6: Error.....	160
506	Figure 15-7: Types	162
507	Figure 15-8: Primitive Types	173
508	Figure 15-9: SelfDevice	175
509	Figure 15-10: EndDevice	176
510	Figure 15-11: FunctionSetAssignments	178
511	Figure 15-12: Pub-Sub	179
512	Figure 15-13: Response	181
513	Figure 15-14: Time.....	183
514	Figure 15-15: DeviceInformation.....	184
515	Figure 15-16: PowerStatus	187
516	Figure 15-17: NetworkStatus	189
517	Figure 15-18: LogEvents	194
518	Figure 15-19: Configuration.....	196
519	Figure 15-20: Files.....	198
520	Figure 15-21: DRLC Event	201
521	Figure 15-22: Load Shed Availability.....	202
522	Figure 15-23: Metering Data	205
523	Figure 15-24: Metering Data Types	206
524	Figure 15-25: Metering Mirror	209
525	Figure 15-26: Metering Mirror Inheritance	210
526	Figure 15-27: Pricing	212
527	Figure 15-28: TextMessage.....	216
528	Figure 15-29: Billing	218
529	Figure 15-30: Billing Readings.....	219
530	Figure 15-31: Prepayment	223
531	Figure 15-32: FlowReservation	228
532	Figure 15-33: DER Info	230
533	Figure 15-34: DER Info Types.....	231
534	Figure 15-35: DER Program.....	231
535	Figure 15-36: DER Curves	232
536	Figure 15-37: DER Control	233
537	Figure 15-38: DER Control Types	234
538	Figure 16-1: Remote Registration.....	252

539	Figure 16-2: Registration Local	255
540	Figure 16-3: Discovery: Function Set Assignment	258
541	Figure 16-4: Discovery Without Function Set Assignment	260
542	Figure 16-5: Discovery Undirected Without Function Set Assignment	262
543	Figure 16-6: Subscription / Notification	263
544	Figure 16-7: Demand Response General	266
545	Figure 16-8: Demand Response - Cancel	270
546	Figure 16-9: Distributed Energy Resource: General	272
547	Figure 16-10: Meter Reading	277
548	Figure 16-11: Metering Interval.....	282
549	Figure 16-12: Metering Instantaneous	287
550	Figure 16-13: Meter Mirroring	291
551	Figure 16-14: Pricing Time of Use.....	304
552	Figure 16-15: Billing Period.....	309
553	Figure 16-16: Billing Historical	312
554	Figure 16-17: Billing Projection.....	315
555	Figure 16-18: File Load - FlowDiagram	317
556	Figure 16-19: Flow Reservation: General	320
557	Figure 16-20: Flow Reservation: Cancel	324
558	Figure 16-21: Simple Event - No Ramp Up - No Ramp Down	328
559	Figure 16-22: Event with Positive Randomization Duration.....	329
560	Figure 16-23: Event with Positive Randomization Start	330
561	Figure 16-24: Event with Negative Randomized Start.....	330
562	Figure 16-25: Positive Randomization in an Hour	331
563	Figure 16-26: Event with Negative Start in One Hour	331
564	Figure 16-27: Randomized Start and Long Ramp Up.....	332
565	Figure 16-28: Randomized Start and Fixed End Time	333
566		

567 1 Introduction

568 1.1 Purpose

569 The purpose of this document is to define the application protocol used by the Smart Energy Profile
570 release 2.0. The Smart Energy Profile Application Protocol 2.0 is designed to meet the requirements
571 stated in the Smart Energy Profile 2.0 Marketing Requirements Document (SEP 2 MRD) [ZB 09-5162]
572 and the Smart Energy Profile 2.0 Technical Requirements Document (SEP 2 TRD) [ZB 09-5449]. Per
573 *Req[DataModel-1]*, this application protocol is an IEC 61968 common information model [61968]
574 profile, mapping directly where possible, and using subsets and extensions where needed, and follows a
575 RESTful architecture [REST].

576 1.2 Scope

577 With respect to the OSI network model, the Smart Energy Profile 2.0 Application Protocol is built using
578 the four layer Internet stack model. This specification defines the 'APPLICATION' layer with TCP/IP
579 providing functions in the 'TRANSPORT' and 'INTERNET' layers. Depending on the physical layer in
580 use (e.g., 802.15.4, 802.11, 1901), a variety of lower layer protocols may be involved in providing a
581 complete solution. Generally, lower layer protocols are not discussed in this document except where there
582 is a direct interaction with the application protocol. The scope of this document is defining the
583 mechanisms for exchanging application messages, the exact messages exchanged including error
584 messages, and the security features used to protect the application messages.

585 1.3 Context Overview

586 The Smart Energy Profile 2.0 Application Protocol is proposed to the ZigBee Alliance and the HomePlug
587 Powerline Alliance to meet the needs specified in the SEP 2 MRD and TRD. The Smart Energy Profile
588 2.0 has been identified as a "standard for implementation" in NIST's Framework and Roadmap for Smart
589 Grid Interoperability Standards [NIST 1108]. As such, this document may be useful for anyone
590 developing a Smart Grid solution.

591 All statements and descriptions made in this document on Smart Energy Profile, SEP, or SE are
592 references to the Smart Energy Profile 2.0 Application Profile.

593 1.4 Document Organization

594 The following documents comprise the definition of SEP 2 and all SEP 2 devices will be required to
595 maintain compliance to these documents:

- 596 • SEP 2 Application Protocol Specification (this document)
- 597 • SEP 2 XML Schema Definition (XSD) (sep.xsd in [ZB 13-0201])
- 598 • SEP 2 WADL (sep_wadl.xml in [ZB 13-0201])

600 The SEP 2 XSD contains the definitions of the SEP 2 resources, attributes, and elements as well as their
601 textual descriptions. The SEP 2 WADL contains the recommended URI structures and use of HTTP
602 methods associated with these objects. In addition a SEP 2 UML model (also contained in
603 [ZB 13-0201]) has been utilized in the creation of the SEP 2 XSD. The SEP 2 UML model is an

604 informative document. Informative textual extracts of the SEP 2 XSD and the SEP 2 WADL are
605 contained in this document for convenience.

606 **1.5 Requirement Language**

607 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
608 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document, when capitalized
609 and in a normative section, constitute normative text and are to be interpreted as described in [RFC 2119].

610 **1.6 Typography Conventions Used**

611 Example URIs, protocol requests, protocol responses, and XML are presented in fixed-width Courier New
612 font, with a size of 8.

613 **1.7 Design Principles**

614 As per the ZigBee Smart Energy Profile 2.0 Technical Requirements Document (TRD) [ZB 09-5449]
615 Section 10.1, Smart Energy Profile 2.0 will follow a RESTful architecture [REST]. The TRD lists a large
616 number of general design principles. Several specific ones are important for the application protocol:

- 617
 - While devices MAY maintain state, interfaces SHOULD be stateless.
 - URI structure SHOULD be clear but as efficient as possible.
 - Minimize the number of transactions required to achieve a given function.

620 2 Acknowledgements

621 On behalf of the ZigBee Alliance, the HomePlug Powerline Alliance, and all of our liaison partners, we
622 would like to extend our thanks to all who participated in the development of this specification.

623 Representatives from many organizations have participated in weekly calls, face-to-face meetings, email
624 exchanges, and specialized editing sessions. Without the broad support from both member and external
625 organizations that care about the development of smart energy solutions, this work would not have been
626 possible. Select individuals stand out for their contributions to this document.

627 When this document was released, the Smart Energy Profile 2 Working Group leadership was composed
628 of the following members:

629 **Robby Simpson:** Chair

630 **Tim Gillman:** Vice-chair

631 **Jeff Shudark:** Secretary

632 **Don Sturek:** Program Manager

633 **Tobin Richardson:** ZigBee Alliance Chairman and CEO

634 **Steven Van Ausdall:** Model, XSD, and WADL Editor

635 **Al Petrick:** Application Specification Editor

636 Our special thanks to the following members who contributed to the document:

637 Casey Anderson	651 Ed Eckert	665 Ricardo Montano
638 Skip Ashton	652 Gene Falendysz	666 Tim Moses
639 Dan Bailey	653 Matthew Gillmore	667 Leslie Mulder
640 Gary Birk	654 Tom Herbst	668 Don Olsen
641 Donald Bowen	655 Lance Hester	669 Ivan O'Neill
642 Doug Burman	656 Ken Holbrook	670 Barry Peirce
643 Paul Carreon	657 Shawn Hu	671 Joseph Reddy
644 Stephen Chasko	658 Rajagopal Iyengar	672 Emil Romascanu
645 Michael Cowan	659 Jeffrey King	673 David Smith
646 Robert Cragie	660 David Kravitz	674 Michael StJohns
647 Wayne Dennison	661 Chris Leclercq	675 Michael Stuber
648 Yusuke Doi	662 Kerry Lynn	676 Mark Ward
649 Jeff Drake	663 Zahra Makoui	
650 Paul Duffy	664 Tony Mauro	

677

678

3 Document Revision History

679

Revision	Version	Description
00	2.0	Smart Energy Profile 2

680 4 References

681 External references in this document are tagged with a document reference identifier placed inline within
682 the text. The references section maps these identifiers to document titles and provides hyperlinks to where
683 the source document may be obtained. Please note, not all referenced documents are freely available.
684 Selected documents may require a membership, subscription, or fee to obtain.

685 4.1 Smart Energy 2.0 Documents

- 686 [ZB 09-5162] ZigBee + HomePlug Joint Working Group Smart Energy Profile Market
687 Requirements Document ([09-5162](#)).
688 <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/HomePlugMarketingRequirementDocument.aspx>
- 690 [ZB 09-5449] Smart Energy Profile Technical Requirements Document ([09-5449](#)).
691 <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/20TechnicalRequirementsDoc.aspx>
- 693 [ZB 13-0201] Smart Energy 2.0 UML Model ([13-0201](#)).

694 4.2 IEC Documents

- 695 [61850] Communication networks and systems in substations - ALL PARTS
696 (<http://webstore.iec.ch>).
- 697 [61968] Application integration at electric utilities - System interfaces for distribution
698 management – ALL PARTS (<http://webstore.iec.ch>).

699 4.3 IETF Documents

- 700 [RFC 768] User Datagram Protocol (<http://tools.ietf.org/html/rfc768>).
- 701 [RFC 793] Transmission Control Protocol (<http://tools.ietf.org/html/rfc793>).
- 702 [RFC 1630] Universal Resource Identifiers in WWW (<http://tools.ietf.org/html/rfc1630>).
- 703 [RFC 1738] Uniform Resource Locators (<http://tools.ietf.org/html/rfc1738>).
- 704 [RFC 1808] Relative Uniform Resource Locators (<http://tools.ietf.org/html/rfc1808>).
- 705 [RFC 2119] Key words for use in RFCs to Indicate Requirement Levels
706 (<http://tools.ietf.org/html/rfc2119>).
- 707 [RFC 2560] X.509 Internet Public Key Infrastructure Online Certificate Status Protocol –
708 OCSP, IETF, M. Meyers et al, June 1999, (<http://tools.ietf.org/html/rfc2560>).
- 709 [RFC 2616] Hypertext Transfer Protocol - HTTP/1.1 (<http://tools.ietf.org/html/rfc2616>).
- 710 [RFC 2717] Registration Procedures for URL Scheme Names
711 (<http://tools.ietf.org/html/rfc2717>).
- 712 [RFC 2818] HTTP Over TLS (<http://tools.ietf.org/html/rfc2818>).
- 713 [RFC 2863] The Interface MIB (<https://tools.ietf.org/html/rfc2863>).
- 714 [RFC 3535] IAB Network Management Workshop (<http://tools.ietf.org/html/rfc3535>).

- 715 [RFC 3635]Definitions of Management Objects for the Internet-Like Interface Types
 716 (<http://tools.ietf.org/html/rfc3625>).
- 717 [RFC 4108]Using Cryptographic Message Syntax (CMS) to Protect Firmware,
 718 IETF, R. Housley, April 2005, (<http://tools.ietf.org/html/rfc4108>).
- 719 [RFC 4193]Unique IPv6 Unicast Addresses (<http://tools.ietf.org/html/rfc4193>).
- 720 [RFC 4291]IP Version 6 Addressing Architecture (<http://tools.ietf.org/html/rfc4291>).
- 721 [RFC 4293]Management Information Base for the Internet Protocol (IP)
 722 (<http://tools.ietf.org/html/rfc4293>).
- 723 [RFC 5246]Transport Layer Security (TLS) Protocol V1.2
 724 (<http://tools.ietf.org/html/rfc5246>).
- 725 [RFC 5280]Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation
 726 List (CRL) Profile, IETF, D Cooper et al, May 2008,
 727 (<http://tools.ietf.org/html/rfc5280>).
- 728 [RFC 5480]Elliptic Curve Cryptography Subject Public Key Information, IETF, S. Turner
 729 et al., March 2009, (<http://tools.ietf.org/html/rfc5480>).
- 730 [RFC 6550]RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks
 731 (<http://tools.ietf.org/html/rfc6550>).
- 732 [RFC 6554]An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-
 733 Power and Lossy Networks (RPL) (<http://tools.ietf.org/html/rfc6554>).
- 734 [RFC 6762]Multicast DNS (<http://tools.ietf.org/html/rfc6762>).
- 735 [RFC 6763]DNS Based Service Discovery (<http://tools.ietf.org/html/rfc6763>).
- 736 [I-D AESCCM]AES-CCM ECC Cipher Suites for TLS (<https://docs.zigbee.org/zigbee-docs/dcn/12/docs-12-0462-00-SEP 2-aes-ccm.pdf>).
- 737 [I-D XMDNS]Extended Multicast DNS (<https://docs.zigbee.org/zigbee-docs/dcn/12/docs-12-0461-00-SEP 2-xmdns.pdf>).

740 4.4 Other References

- 741 [ASN.1]ITU-T Recommendation X.680 – Information technology – Abstract Syntax
 742 Notation One (ASN.1): Specification of basic notation, ITU, July 2002.
- 743 [CSEP]Consortium for SEP 2 Interoperability (<http://www.csep.org>).
- 744 [DOCSIS]DOCSIS (<http://cablelabs.com/cablemodem/specifications/index.html>).
- 745 [EPRI]Common Functions for Smart Inverters. EPRI, Palo Alto, CA: 2011. Product ID
 746 Number 1023059
 747 (<http://my.epri.com/portal/server.pt?space=CommunityPage&cached=true&parentname=ObjMgr&parentid=2&control=SetCommunity&CommunityID=221&PageIDqueryComId=0>).
- 750 [EXI]Efficient XML Interchange (EXI) Format 1.0 (<http://www.w3.org/TR/2008/WD-exi-20080919/>).

- 752 [EXIBP]Efficient XML Interchange (EXI) best practices (<http://www.w3.org/TR/exi-best-practices/>).
753
- 754 [Fielding]Fielding, Roy Thomas (2000), "Architectural Styles and the Design of Network-based Software Architectures", Doctoral Dissertation, University of California, Irvine (<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.html>).
755
756
- 757 [IEEE 802.1AR].....Standard for Local and Metropolitan Area Networks: Secure Device Identity, IEEE, 2009 (<http://www.ieee802.org/1/pages/802.1ar.html>).
758
- 759 [ISO 4217]Codes for the representation of currencies and funds
760 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=46121).
761
- 762 [Linux]Linux (<https://www.linux.com/>).
763
- 764 [NIST-1108]NIST framework and roadmap for Smart Grid Interoperability Standards, Release 1.0.
765 (http://collaborate.nist.gov/twisisgrid/pub/SmartGrid/IKBFramework/NISTFrameworkAndRoadmapForSmartGridInteroperability_Release1final.pdf).
766
- 767 [NIST SP800-131A]Transition: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths (<http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>).
768
769
- 770 [NIST SP800-131B]....Transition: Validation of Transitioning Cryptographic Algorithm and Key Lengths (http://csrc.nist.gov/publications/drafts/800-131B/draft-SP800-131B_February2011.pdf).
771
772
- 773
- 774 [NIST SP800-131C].....Transitions: Validating the Transition from FIPS 1862 to FIPS 1863
775 (http://csrc.nist.gov/publications/drafts/800-131C/draft-SP800-131C_February2011.pdf).
776
777
- 778 [PEN]Internet Authority Assignment numbers (IANA) PEN
779 (<http://pen.iana.org/pen/PenApplication.page>).
780
- 781 [REST]Representational State Transfer
782 (http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm).
783
- 784 [SunSpec].....SunSpec Alliance Inverter Control Model, work in progress.
785 (<http://www.sunspec.org/specifications/>).
786
- 787
- 788 [WADL].....Web Application Description Language (<http://www.w3.org/Submission/wadl/>).
789 [XML]Extensible Markup Language (<http://www.w3.org/TR/REC-xml/>).
790

790 5 Definitions, Acronyms and Abbreviations**791 5.1 Acronyms and Abbreviations**

792 This subsection provides a list of acronyms and abbreviations introduced in this document. For a
793 comprehensive treatment of acronyms and abbreviations used, please refer to Section 4 of the Smart
794 Energy Profile Technical Requirements Document [ZB 09-5449].

- 795 ACL Access Control List
796 CA..... Certificate Authority
797 CRL..... Certificate Revocation List
798 CSEP..... Consortium for SEP 2 Interoperability
799 DER Distributed Energy Resource
800 DNS Domain Name System
801 DNS-SD DNS-based Service Discovery
802 DRLC..... Demand Response and Load Control
803 ECDSA Elliptic Curve Digital Signature Algorithm
804 EMS Energy Management System
805 ESI Energy Services Interface
806 EVSE Electric Vehicle Supply Equipment
807 EXI..... Efficient XML Interchange
808 HTTP Hypertext Transfer Protocol
809 IETF Internet Engineering Task Force
810 IP Internet Protocol
811 LFDI Long Form Device Identifier
812 MCA Manufacturer's CA
813 MICA Manufacturing Issuing CA
814 OCSP Online Certificate Status Protocol
815 OID Object Identifier
816 PKI..... Public Key Infrastructure
817 REST..... Representational State Transfer
818 SERCA..... Smart Energy Root CA
819 SFDI..... Short Form Device Identifier
820 TCP Transmission Control Protocol
821 UOM Units of Measure
822 URI Uniform Resource Identifier
823 XML Extensible Markup Language

824 WADL Web Application Description Language

825 **5.2 Definitions**

826 A limited set of terms specific to this application specification are introduced below. For a
827 comprehensive treatment of the terms used, please refer to Section 4 of the Smart Energy Profile
828 Technical Requirements Document [ZB 09-5449].

829 ACCESS CONTROL LIST

830 A security mechanism in which entities and authorizations (e.g., read, write, create, delete)
831 are related to resources to determine the entities' allowed operations on the resources.

832 CERTIFICATE AUTHORITY

833 An entity that issues digital certificates for use by other entities.

834 CERTIFICATE CHAIN

835 A chain of certificates, with each certificate's signature verified using the key from the next
836 certificate in the chain. The single exception is the certificate at the end of the chain (the
837 trust anchor), known as the root CA certificate, which is self signed.

838 CLIENT

839 The device or host that interacts with a server to obtain information related to a resource
840 hosted by the server.

841 DEVICE CERTIFICATE

842 A digital certificate installed within a device that binds the device identity to the device.
843 Device Certificates are exchanged by network access control and application protocols to
844 authenticate devices as genuine SEP 2 and further to prove specific device identity.

845 ENERGY SERVICES INTERFACE (ESI)

846 A device, with multiple network interfaces, which is a member of both the home smart
847 energy network and a service provider's private network. This is the primary mechanism for
848 the service provider to contribute data and directives into the smart energy network and to
849 receive responses from smart energy devices.

850 FINGERPRINT

851 This is the result of summarizing a certificate with a secure hash function. The fingerprint is
852 generally expressed as a hex string. It is used to confirm the integrity of a certificate
853 obtained over an untrusted channel.

854 FUNCTION SET

855 A logical grouping of resources that cooperate to implement SEP 2 features (e.g., Metering,
856 Demand Response and Load Control).

857 FUNCTION SET ASSIGNMENTS

858 A logical addressing mechanism in SEP 2 that allows devices to be directed to use specific
859 resources (e.g., to facilitate a device's participation in a program). Please see Section 10.5
860 for details.

861 HOST

862 This is the representation of a device in its application context. Typically represented by an
863 IP address or domain name.

864 **INTERMEDIATE CA**

865 A CA below the root CA that issues certificates to subordinate CA's.

866 **ISSUING CA**

867 A CA that issues certificates to devices or code-signers.

868 **MANUFACTURER'S CA (MCA)**

869 An Intermediate CA operated by a specific manufacturer for the purpose of issuing MICA's
870 for that manufacturer.

871 **MANUFACTURING ISSUING CA (MICA)**

872 An Issuing CA that issues certificates to devices during the manufacturing process.

873 **MANUFACTURING PKI**

874 The set of CAs that issue certificates to devices during the manufacturing process. The set
875 includes the Smart Energy Root CA, Manufacturer's CA's and Manufacturing Issuing CA's.

876 **NODE**

877 This is the representation of a device in its network context, typically represented as an IP
878 address.

879 **OID**

880 Object identifier. An OID consists of a node in a hierarchically-assigned namespace,
881 formally defined using the ITU-T's ASN.1 standard [ASN.1].

882 **PEM FORMAT CERTIFICATE**

883 An X.509 certificate that has been Base64 encoded and wrapped in "-----BEGIN
884 CERTIFICATE-----","-----END CERTIFICATE-----"sentinels for transport as a text file or
885 block.

886 **PKI**

887 Public Key Infrastructure. A set of hardware, software, people, policies, and procedures
888 needed to create, manage, distribute, use, store, and revoke digital certificates.

889 **REGISTERED**

890 The state of a device with regards to a particular server wherein an EndDevice record for the
891 device has been populated on the server and that record contains a valid registration record.
892 These records are typically populated with device information transmitted out-of-band to the
893 server's owner.

894 **RESOURCE**

895 URI addressable object that is manipulated via the RESTful uniform interface.

896 **RESOURCE DISCOVERY**

897 The process whereby Clients identify Resources being served on the network. Clients issue
898 a request to all devices on the network requesting resource(s) of interest. Servers hosting the
899 requested resource(s) respond with information necessary to access the Server and its
900 resource(s).

901	ROOT CA
902	A CA whose certificate or public key is a trust anchor for any other certificates in a chain of trust.
904	ROOT CERTIFICATE
905	The Root CA's self-signed certificate. Generally also a Trust Anchor.
906	SELF-SIGNED CERTIFICATE
907	A certificate whose issuer and subject are identical, and whose public key verifies its signature.
909	SERVER
910	The device or host that holds a resource and exposes representations of that resource.
911	SMART ENERGY ROOT CA (SERCA)
912	The top-level CA for the SEP 2 Manufacturing PKI.
913	TRUST ANCHOR
914	The root of trust for a certificate chain. This is an authoritative entity represented by a public key and associated data and is generally provided in the SEP 2 hierarchy in the form of a self-signed certificate.
917	TRUSTED ROOT STORE
918	An integrity-protected location for storing root certificates.

919 6 Design Pattern

920 6.1 Protocol Flexibility

921 The Smart Energy Profile is designed to implement a REST architecture. It is built around the core
922 actions of GET, HEAD, PUT, POST, and DELETE (as used in [REST]), with the addition of a
923 lightweight subscription mechanism as discussed in Section 10.6. Any application protocol that can
924 implement a RESTful command set could likely be used with SEP 2, but HTTP is a required baseline
925 for interoperable SEP 2 implementations. HTTP utilizes TCP as its transport protocol. As a result, TCP
926 manages the session providing delivery assurance and windowing.

927 Smart Energy Profile 2.0 servers and clients SHALL be compliant with [RFC 2616].

928 6.2 General Rules / Best Practices

929 This specification shall not make distinctions between servers, clients, or devices, when defining
930 interfaces and URIs. The goal is to avoid having a resource that has one behavior on a server and a
931 different behavior on a client or different type of server. The distinction between the server and client
932 role depends on whether a device exposes a resource (server) or interacts with the resource (client).

933 The default mechanism for obtaining a resource representation is a pull mechanism, implemented with
934 GET. A client requests and retrieves data from a server or creates, modifies, or deletes data on a server.

935 The use of a subscription mechanism for retrieving a resource representation is also optionally
936 supported, where convenient and appropriate. Resources that support subscription are denoted by their
937 subscribable attribute.

938 Objects have a defined granularity and whole objects (not partial objects) are to be updated with the
939 granularity defined in the schema.

940 Clients that expect to have intermittent connections to the network (e.g., battery-powered sleepy devices,
941 mobile devices, etc.) use a pull mechanism as their default behavior for resource retrieval, as a
942 subscription / notification mechanism may not be reliable. It should be noted that clients that expect to
943 have intermittent connections to the network may still POST, PUT, and DELETE resources, provided
944 they have the appropriate security permissions.

945 The TCP ports used for HTTP or HTTPS SHALL be specified in the xmDNS service advertisement for
946 the service.

947 Content SHALL be transferred with either one of the content types: "application/sep+xml" or
948 "application/sep-exi".

949 Devices do not assume the use of the URIs and their structures given throughout this specification. All
950 resources are self-describing as it is acknowledged that URIs, schemas, and resources might change in
951 the future. All resources SHALL contain links to their subordinate resources to support flexibility in
952 URIs and future extensibility. Thus, to allow for extensibility and granularity, all objects are described
953 in schemas and referenced via URIs. The URIs presented throughout this specification are
954 recommendations. Thus, clients do not assume that URIs for resources are fixed on all servers or even
955 on a given server (over time), but rather retrieve the appropriate URIs through resource discovery and
956 links within resources. For network efficiency, devices MAY assume URIs are fixed on a particular
957 server over time. If a URI returns an unexpected result, the client SHOULD execute resource discovery
958 to determine the new URI value.

959 Version information should not be presented in the URI unless that version information is inherent to the
 960 name of that resource. If necessary and for reasons of extensibility, version information is provided
 961 within the associated resources and / or schemas.

962 All values in XML and EXI SHOULD be represented as compactly as possible. Decimal values
 963 SHOULD be represented without leading zeros. Hexadecimal values SHALL be represented with one
 964 leading zero, if needed, to ensure an even number of digits.

965 6.3 **WADL**

966 The SEP 2 RESTful interface is defined using the Web Application Description Language (WADL)
 967 [ZB 13-0201].

968 Smart Energy Profile 2.0 devices SHALL conform to the interface specifications contained in the
 969 WADL as follows.

- 970 • Devices SHALL conform to the WADL specification as per [WADL].
- 971 • Devices SHALL conform to the WADL definition in [ZB 13-0201]. By implication, all resource
 972 representations SHALL validate per the schema [ZB 13-0201] within the standardized SEP
 973 XML namespace (<http://zigbee.org/sep>).
- 974 • Compliance (MODE) designations are interpreted as follows.
 - 975 ○ Mandatory ("M") - Devices MUST implement and conform.
 - 976 ○ Optional ("O") - Devices MAY implement, and if implemented, MUST conform.
 - 977 ○ Discouraged ("D") - Devices SHOULD NOT implement, but if implemented, MUST
 978 conform.
 - 979 ○ Error ("E") - Devices MUST return one of the specified response status codes (e.g. 400
 980 – Bad Request or 405 – Method Not Allowed).

981 6.4 **Schema**

- 982 • Resources located at URIs returned in the href attribute of "Link" specializations (e.g.,
 983 EndDeviceListLink, SelfDeviceLink) SHALL conform to the schema definition for that object,
 984 which is the tag name with the "Link" suffix removed. For example, the resource at
 985 SelfDeviceLink href follows the definition for the SelfDevice resource.
- 986 • If a client PUTs or POSTs a resource to a server containing attributes or elements that instead
 987 are to be populated by the server (e.g., href), the server SHALL return an HTTP 400 error.
- 988 • If a function set is not implemented, Link elements to resources in that function set SHALL
 989 NOT be included.

990 6.5 **Uniform Resource Identifiers**

991 HTTP uses ASCII text for transferring URIs between clients and servers, as well as for including
 992 options and details regarding the message content. These transfers occur with every transaction. If the
 993 naming scheme used for URIs is overly verbose, these transactions become needlessly inefficient on
 994 constrained networks. Of course, if the naming scheme used is overly cryptic, the advantages of a text-
 995 based protocol are lost, and it becomes difficult for someone troubleshooting to decipher a transaction.

996 The following conventions are used for URI naming:

- 997 1) URI elements are at most 4 characters, but recognizable to a knowledgeable engineer.
 998 Element names as short as 1 character are acceptable provided their meaning is clear.
- 999 2) URI elements are constructed of consonants only, unless inclusion of a vowel adds clarity,
 1000 such as a leading vowel or well-known abbreviation.

- 1001 3) URI elements are in all lower case.
1002 4) URIs SHALL NOT be greater than 255-bytes in length. In practice, URIs SHOULD be
1003 much smaller than 80-bytes.

1004 **6.6 List Resources**

1005 Many resources within this specification are derived from the <List> object. Throughout this
1006 specification, these resources will collectively be referred to as list resources.

1007 The following attributes are defined for list resources:

- 1008 • `all` – used to indicate the total number of items (subordinate resources) that exist in the list
1009 resource. This number may vary according to the client's access privileges.
1010 • `results` – used to indicate the number of items (subordinate resources) included in a specific
1011 subset of the list (result from a paged GET query to the list, etc.). This value will always be less
1012 than or equal to 'all'.

1013 Clients and servers use these attributes, combined with the query string parameters described below, to
1014 implement paged access to lists. Client control of list paging is important for resource-constrained
1015 devices.

1016 List items (subordinate resources) are read using one of the two idioms described below:

- 1017 1) Ordinal access to the first, second, nth, etc. item in the list is supported via query string
1018 parameters included with a GET to the list resource URI.
1019 2) Random access to specific list items is supported via a GET directly to the URI of the list item.

1020 Some list items may be written by POSTing a list item representation to the URI of the list. (e.g.,
1021 notifications), while others may be created using private interfaces over the provider network. Ordinal
1022 placement of the new resource within the list is determined by the list sort order (defined by each list).

1023 Each function set defined in this specification that contains list resources includes a List Ordering table.
1024 Each list resource has an entry in the corresponding table that describes the details of one or more
1025 unique sort keys and the precedence of those keys. A list resource's elements SHALL be ordered
1026 according to this specified List Ordering.

1027 **6.6.1 Query String Parameters**

1028 Query string parameters are parameters added to a URI to provide filtering / paging of list items returned
1029 in query results.

1030 The list paging mechanism allows GET requests to specify the range of list items to be returned in a
1031 query result set. The general syntax of a paged query is as follows:

1032 `{URI}?s={x}&a={y}&l={z}`

1033 Where `{URI}` represents a URI used to address a list resource, `(s | a | l)` represent query string
1034 parameters (further defined below), and `{x}`, `{y}`, and `{z}` represent the respective query string
1035 parameter values.

1036 The query string parameters are defined as:

- 1037 • `s` – ("start") is used to indicate the first ordinal position in the list to be returned in the query
1038 result list as determined by the list's ordering. The value is specified in decimal. The first ordinal
1039 position of the list SHALL be designated with a value of '0' and the maximum possible value is
1040 '65535'. If this query string parameter is not specified, the default start value SHALL be '0'.

- `a` – ("after") is used to indicate that only items whose primary key occurs after the given date / time parameter should be included in the query result list. This query string parameter is only applied to list resources that are ordered using a time-based primary key. The parameter SHALL be ignored if the primary key is not time-based. The format of the parameter SHALL be a 64-bit decimal number with identical semantics as that of the `TimeType` (see Section 12.2 and the XML XSD in [ZB 13-0201]).
- `l` – ("limit") is used to set the maximum number of list items (up to 255) to be included in the query result list. The value is specified in decimal. If this query string parameter is not specified, the default limit SHALL be '1'. Servers MAY return a result list smaller than that specified by the client provided limit.

If both a "start" and "after" query string parameter are used simultaneously, the "after" query string parameter SHALL have precedence. The "start" position 0 SHALL be relative to the position specified by the "after" parameter.

If a query string requests a list element that does not exist (e.g., `s=3` when there are two items in the list), servers SHALL return an empty list representation.

Readers should note that the "after" query string parameter SHOULD NOT be used alone for paging through a list. As some list resources MAY contain multiple subordinate resources with the same time-based primary key, it is RECOMMENDED that clients wishing to paginate a list resource while using the "after" query string parameter should keep the value for the "after" query string parameter constant while changing the "start" query string parameter.

If a particular query string parameter appears more than once, then the first occurrence of the query string parameter SHALL be used (in left-to-right order) and subsequent occurrences MUST be ignored.

Server receipt of a query parameter unknown to the server MUST be ignored by the server and MUST NOT generate an HTTP error. Servers MUST NOT generate resource representations containing `href` attributes that contain query parameters. Clients MUST ignore query parameters contained in resource `hrefs`, but SHOULD NOT remove them if the URI is used for subsequent RESTful exchanges.

Should an empty list representation be requested (either through the use of query string parameters such as `l=0` or when the list itself is empty), the server SHALL return no subordinate representations, but SHALL return any other elements that may be defined for the list.

Clients MUST NOT assume any index semantics for list URIs. For example, a client desiring to read the 3rd item from the list at `http://some-host/somelist` MUST NOT assume a GET to `http://some-host/somelist/2` will return the third item. The correct access is supported by the client issuing a GET to `http://some-host/somelist?s=2&l=1`.

The following examples demonstrate the use of query string parameters with a list resource. Consider the `MyTypeList` resource, depicted below:

```

1076 <MyTypeList href="http://host1/the/list" all="7" results="7">
1077   <MyType href="http://host1/instance/of/type/red">
1078     <timeStamp>100</timeStamp>
1079   </MyType>
1080   <MyType href="http://host2/instance/of/type/green">
1081     <timeStamp>200</timeStamp>
1082   </MyType>
1083   <MyType href="http://host3/instance/of/type/blue">
1084     <timeStamp>300</timeStamp>
1085   </MyType>
1086   <MyType href="http://host4/instance/of/type/yellow">
1087     <timeStamp>400</timeStamp>
1088   </MyType>
1089   <MyType href="http://host5/instance/of/type/black">
1090     <timeStamp>500</timeStamp>
1091   </MyType>

```

```

1092      <MyType href="http://host6/instance/of/type/white">
1093          <timeStamp>600</timeStamp>
1094      </MyType>
1095      <MyType href="http://host7/instance/of/type/orange">
1096          <timeStamp>700</timeStamp>
1097      </MyType>
1098  </MyTypeList>
1099

```

1100 This list is sorted in ascending order per the `<timeStamp/>` element of `MyType`.

1101 A GET to `http://host1/the/list?s=0&l=1` will return:

```

1102
1103 <MyTypeList href="http://host1/the/list" all="7" results="1">
1104     <MyType href="http://host1/instance/of/type/red">
1105         <timeStamp>100</timeStamp>
1106     </MyType>
1107 </MyTypeList>
1108

```

1109 A GET to `http://host1/the/list?s=0&l=5` will return:

```

1110
1111 <MyTypeList href="http://host1/the/list" all="7" results="5">
1112     <MyType href="http://host1/instance/of/type/red">
1113         <timeStamp>100</timeStamp>
1114     </MyType>
1115     <MyType href="http://host2/instance/of/type/green">
1116         <timeStamp>200</timeStamp>
1117     </MyType>
1118     <MyType href="http://host3/instance/of/type/blue">
1119         <timeStamp>300</timeStamp>
1120     </MyType>
1121     <MyType href="http://host4/instance/of/type/yellow">
1122         <timeStamp>400</timeStamp>
1123     </MyType>
1124     <MyType href="http://host5/instance/of/type/black">
1125         <timeStamp>500</timeStamp>
1126     </MyType>
1127 </MyTypeList>
1128

```

1129 A GET to `http://host1/the/list?s=5&l=1` will return:

```

1130
1131 <MyTypeList href="http://host1/the/list" all="7" results="1">
1132     <MyType href="http://host6/instance/of/type/white">
1133         <timeStamp>600</timeStamp>
1134     </MyType>
1135 </MyTypeList>
1136

```

1137 A GET to `http://host1/the/list?s=5&l=5` will return:

```

1138
1139 <MyTypeList href="http://host1/the/list" all="7" results="2">
1140     <MyType href="http://host6/instance/of/type/white">
1141         <timeStamp>600</timeStamp>
1142     </MyType>
1143     <MyType href="http://host7/instance/of/type/orange">
1144         <timeStamp>700</timeStamp>
1145     </MyType>
1146 </MyTypeList>
1147

```

1148 A GET to `http://host1/the/list?s=12&l=2` will return a list representation containing 0 items:

```

1149
1150 <MyTypeList href="http://host1/the/list" all="7" results="0">
1151 </MyTypeList>
1152

```

1153 or <MyTypeList href="http://host1/the/list" all="7" results="0" />

1154
1155

1156 A GET to `http://host6/instance/of/type/white` will return:

```
1157 <MyType href="http://host6/instance/of/type/white">
1158     <timeStamp>600</timeStamp>
1159 </MyType>
1160
```

1161 A GET to `http://host1/the/list?a=400&l=4` will return:

```
1162
1163 <MyTypeList href="http://host1/the/list" all="7" results="3">
1164     <MyType href="http://host5/instance/of/type/black">
1165         <timeStamp>500</timeStamp>
1166     </MyType>
1167     <MyType href="http://host6/instance/of/type/white">
1168         <timeStamp>600</timeStamp>
1169     </MyType>
1170     <MyType href="http://host7/instance/of/type/orange">
1171         <timeStamp>700</timeStamp>
1172     </MyType>
1173 </MyTypeList>
1174
```

1175 A GET to `http://host1/the/list?a=400&s=0&l=2` will return:

```
1176
1177 <MyTypeList href="http://host1/the/list" all="7" results="2">
1178     <MyType href="http://host5/instance/of/type/black">
1179         <timeStamp>500</timeStamp>
1180     </MyType>
1181     <MyType href="http://host6/instance/of/type/white">
1182         <timeStamp>600</timeStamp>
1183     </MyType>
1184 </MyTypeList>
1185
```

1186 A GET to `http://host1/the/list?a=400&s=2&l=2` will return:

```
1187
1188 <MyTypeList href="http://host1/the/list" all="7" results="1">
1189     <MyType href="http://host7/instance/of/type/orange">
1190         <timeStamp>700</timeStamp>
1191     </MyType>
1192 </MyTypeList>
```

1193 6.7 Resource Design Rules

1194 The following rules apply to the design and use of list resources:

- 1195 • List resources SHALL support "start" and "limit" query string parameters, thus always
1196 supporting paging.
- 1197 • List resources that have a time-based primary key SHALL support the "after" query string
1198 parameters.
- 1199 • All subordinate resources of list resources SHALL include an `href` attribute containing the URI
1200 of the subordinate resource.
- 1201 • Each list resource described in this specification includes a List Ordering that defines how the
1202 list is ordered, including the details of one or more keys used to order the list and the precedence
1203 of these keys.
- 1204 • When queried, list resources SHALL return subordinate resources in the order defined by the
1205 List Ordering.
- 1206 • All subordinate resources of list resources that support multiple types, e.g., `NotificationList`,
1207 SHALL include an `xsi:type` attribute. In this case, the XML Schema Instance Namespace must
1208 also be declared. See Section 16 for examples.

1209 The following rules apply to the design and use of non-list resources:

- 1210 • Non-list resources SHALL NOT support the defined query string parameters. Query string
1211 parameters applied to non-list resource URIs SHOULD be ignored.

7 Application Support

7.1 Overview

This section details the standards-based application transport protocol and other lower-layer requirements required for interoperability of SEP 2 devices.

The application support layer provides the following services:

- RESTful HTTP/1.1 as the application data exchange semantics, as described in [RFC 2616] and [Fielding].
- XML [XML] and / or EXI [EXI] encoding as the data payload of the RESTful operations.
- Transport authentication and encryption using TLS over HTTP [RFC 2818] and [RFC 5246].

7.2 Use of TCP

The choice of HTTP/1.1 as the application data exchange protocol leads directly to the use of TCP as the transport protocol. [RFC 2616] states that:

"HTTP communication usually takes place over TCP / IP connections. The default port is TCP 80 [19], but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used; the mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question are outside the scope of this specification."

Hence, albeit that other transport protocols may be used to carry HTTP, they are required to provide reliable transport. Clearly, the de facto choice is TCP [RFC 793].

7.3 URI Encoding

The address of the resources presented by an SEP 2 host will use the standard URI syntax specific to HTTP/1.1 (i.e., `http://`) as per [RFC 1630], [RFC 1738], and [RFC 1808].

It is recommended that SEP 2 use the formally defined and registered (IANA) URN namespace definitions to address its application level resources as referenced in [RFC 2717] and [RFC 1808].

Given the constrained nature of many SEP 2 hosts, it is critical to devise the URL namespace scheme (hierarchy) that is both descriptive and compact. For more information, see Section 6.5.

7.4 HTTP Headers

HTTP/1.1 defines a variety of header types that have the potential to be verbose. As an example, the Accept header is illustrated below showing lengths varying from 35 octets to 106 octets:

- `Accept: audio/*; q=0.2, audio/basic (35 octets)`
- `Accept: text/*; q=0.3, text/html; q=0.7, text/html; level=1, text/html; level=2; q=0.4, */*; q=0.5 (92 octets)`
- `Accept: text/xml, application/xml, application/xhtml+xml, text/html; q=0.9, text/plain; q=0.8, image/png, */*; q=0.5 (106 octets)`

Given the packet size constraints for many SEP 2 networks, the set of HTTP headers supported must be curtailed through best practices recommendations realizing that a host cannot prevent a verbose message from being sent to it.

1250 SEP 2 provides a set of Mandatory, Optional, and Discouraged recommendations for each HTTP header.
 1251 SEP 2 implementations SHOULD employ only Mandatory HTTP headers, minimize the use of Optional
 1252 HTTP headers, and avoid the use of Discouraged headers.

1253 **7.4.1 HTTP Header Field Recommended Usage**

1254 In the following table, the HTTP/1.1 header fields have been annotated with the following labels:

- 1255 • MANDATORY: support for the field is REQUIRED.
 1256 • OPTIONAL: support for this field is left to the implementer's discretion.
 1257 • DISCOURAGED: to conserve code space and / or bandwidth, support for this field, while not
 1258 explicitly forbidden, is not recommended.

1259 The following table summarizes the recommended use of HTTP/1.1 headers in SEP 2:

1260 **Table 7-1 HTTP Headers.**

Header	Used in Message Type		RFC Required / Optional	SEP 2 Use
Accept	Request		Optional	Mandatory
Accept-Charset	Request		Optional	Discouraged
Accept-Encoding	Request		Optional	Discouraged
Accept-Language	Request		Optional	Discouraged
Accept-Ranges	Request	Response	Optional	Discouraged
Age		Response	Optional (Required for a cache)	Discouraged
Allow		Response	Required	Mandatory
Authorization	Request	Response	Optional	Discouraged
Cache-Control	Request	Response	Optional	Discouraged
Connection	Request		Optional (Required in some situations (e.g., HTTP/1.1 applications that do not support persistent connections))	Optional (Mandatory in some situations)
Content-Encoding		Response	Optional (Required when an encoding is applied)	Discouraged
Content-Language		Response	Optional	Discouraged
Content-Length	Request	Response	Optional (Required in many situations, see section 4.4 of [RFC 2616])	Optional (Mandatory in many situations, see section 4.4 of [RFC 2616])
Content-Location		Response	Optional	Discouraged
Content-MD5		Response	Optional	Discouraged

Header	Used in Message Type		RFC Required / Optional	SEP 2 Use
Content-Range		Response	Optional	Optional (see Section 11.7.1.4)
Content-Type	Request	Response	Required	Mandatory
Cookies			Optional	Discouraged
Date	Request	Response	Mandatory	Mandatory
ETag		Response	Optional	Optional (see Section 11.7.1.4)
Expect	Request		Optional	Discouraged
Expires		Response	Optional	Discouraged
From	Request		Optional	Discouraged
Host	Request		Required	Mandatory
If-Match	Request		Optional	Discouraged
If-Modified-Since	Request		Optional	Discouraged
If-None-Match	Request		Optional	Discouraged
If-Range	Request		Optional	Discouraged
If-Unmodified-Since	Request		Optional	Discouraged
Last-Modified		Response	Optional	Discouraged
Location		Response	Optional	Mandatory in many situations (e.g., POST responses)
Max-Forwards	Request		Optional	Discouraged
Pragma	Request	Response	Optional	Discouraged
Proxy-Authenticate		Response	Optional	Discouraged
Proxy-Authorization	Request		Optional	Discouraged
Range	Request		Optional	Optional (see Section 11.7.1.4)
Referrer	Request		Optional	Discouraged
Retry-After		Response	Optional	Optional (see Section 11.7.1.4)
Server		Response	Optional	Discouraged
TE	Request		Discouraged	Discouraged
Trailer		Response	Discouraged	Discouraged

Header	Used in Message Type		RFC Required / Optional	SEP 2 Use
Transfer-Encoding		Response	Optional	Discouraged
Upgrade	Request		Optional	Discouraged
User-Agent	Request		Optional	Discouraged
Vary		Response	Discouraged	Discouraged
Via	Request	Response	Optional	Discouraged
Warning	Request	Response	Discouraged	Discouraged
WWW-Authenticate		Response	Optional	Discouraged

1261 **7.5 HTTP Response Codes**

1262 Response codes are expected to be generalized across RESTful platforms. The specific uses detailed
 1263 below are likely to be generalized. In the interest of clarity and completeness, they are included here.
 1264 Please note that these response codes follow general best practices for RESTful interfaces, though they
 1265 are tuned to address some of the limitations of the embedded space.

1266 This sub-section attempts to highlight HTTP response codes that are felt to be more important or that
 1267 need special attention from developers. However, SEP 2 clients may encounter any HTTP response code
 1268 defined by [RFC 2616] and, all use of, and response to HTTP response codes SHALL be specification
 1269 and RFC compliant.

1270 **7.5.1 Common Responses**

1271 The following HTTP response codes are those considered to be of utmost importance for this
 1272 specification.

1273 **7.5.1.1 1xx (Informational)**

1274 These response codes are informational in purpose and are used to indicate that the server is continuing
 1275 to process in some fashion.

1276 [RFC 2616] states, "A client MUST be prepared to accept one or more 1xx status responses prior to a
 1277 regular response, even if the client does not expect a 100 (Continue) status message. Unexpected 1xx
 1278 status responses MAY be ignored by a user agent."

1279 **7.5.1.2 200 ("OK")**

1280 This response code is sent to indicate a successful transaction.

1281 This response code is often used in response to a successful GET request, with the entity-body
 1282 containing a representation of the requested resource. Use of this response code in response to PUT,
 1283 POST, or DELETE requests is discouraged, to avoid the potentially unnecessary traffic generated by
 1284 returning the resource representation in the entity-body (see 201 ("Created") and 204 ("No Content")).

1285 **7.5.1.3 201 ("Created")**

1286 This response code is sent to indicate a new resource has been created, at the client's request with a PUT
 1287 or POST.

1288 The Location header SHALL be used in conjunction with this response code to indicate the URI of the
 1289 newly created resource. The inclusion of a representation of the newly created resource in the entity-
 1290 body of the response is discouraged, to conserve bandwidth.

1291 [RFC 2616] states, "If a new resource is created, the origin server MUST inform the user agent via the
1292 201 (Created) response."

1293 7.5.1.4 **204 ("No Content")**

1294 This response code is sent to indicate a successful transaction, but one where the response does not
1295 include an entity-body.

1296 This response code is often used in response to a successful PUT or POST request, where the resource is
1297 modified, not created. This response code is also sent in response to a successful DELETE request. This
1298 response code is also sent in response to a successful GET request, where the resource exists but has an
1299 empty representation.

1300 Further, when there are URIs that point to a resource that does not yet have content (an "empty
1301 representation"), this response code SHOULD be returned. For instance, if a client created a new
1302 resource with a POST and that new resource contains URIs pointing to resources that were not yet
1303 created and then a client were to request those linked resources, this response code would be the best
1304 response. When those resources are created (via a PUT, for instance), this response code (204)
1305 SHOULD be returned (in response to the PUT, for instance).

1306 7.5.1.5 **206 ("Partial Content")**

1307 This response code is sent to indicate the server has fulfilled the partial GET request (as specified by the
1308 Range header) for a resource. Note that [RFC 2616] requires the Content-Range and Date headers
1309 MUST be present in the response.

1310 7.5.1.6 **301 ("Moved Permanently")**

1311 This response code is sent to indicate that the requested resource has a new URI. The Location header
1312 SHOULD be used in conjunction with this response code to indicate the new URI of the requested
1313 resource. The entity-body of the response SHOULD be empty. Upon unexpected receipt of this response
1314 code, clients SHOULD perform resource discovery to determine which resources have changed location.

1315 7.5.1.7 **302 ("Redirect")**

1316 The Location header SHALL be used in conjunction with this response code to indicate the new URI of
1317 the requested resource. The entity-body of the response SHOULD be empty.

1318 This response code is often used to redirect URI's requested as HTTP to HTTPS.

1319 7.5.1.8 **400 ("Bad Request")**

1320 This response code is used to indicate a client-side error and is used when no other 4xx response code is
1321 appropriate. Often, this response code indicates that the representation sent by a client with a PUT or
1322 POST is not appropriate or is malformed.

1323 [RFC 2616] states, "All Internet-based HTTP/1.1 servers MUST respond with a 400 (Bad Request)
1324 status code to any HTTP/1.1 request message which lacks a Host header field."

1325 7.5.1.9 **401 ("Unauthorized")**

1326 This response code is used when a client does not have proper authorization to perform the requested
1327 action on a resource.

1328 Note, if a server did not wish a client to know of the existence of the resource, it should instead send a
1329 404 ("Not Found") response code.

1330 [RFC 2616] states, "The response MUST include a WWW-Authenticated header field containing a
1331 challenge applicable to the requested resource."

1332 7.5.1.10 **404 ("Not Found")**

1333 This response code is used to indicate that no resource can be found at the specified URI.

1334 This response code MAY also be used in lieu of a 401 response code.

1335 **7.5.1.11 405 ("Method Not Allowed")**

1336 This response code is used to indicate that the resource does not allow the HTTP method used by the
1337 client.

1338 [RFC 2616] states, "The response MUST include an Allow header containing a list of valid methods for
1339 the requested resource."

1340 **7.5.1.12 406 ("Not Acceptable")**

1341 This response code is used to indicate that a server is unable to generate a response that is acceptable
1342 according to the Accept headers sent in the request.

1343 **7.5.1.13 413 ("Request Entity Too Large")**

1344 This response code is used to indicate that a server is refusing to process a request, as the request is
1345 larger than the server is willing or able to process.

1346 **7.5.1.14 416 ("Requested Range Not Satisfiable")**

1347 This response code is used to indicate that a server has received a Range request that does not overlap
1348 any of the resource content.

1349 [RFC 2616] states "A server sending a response with status code 416 (Requested range not satisfiable)
1350 SHOULD include a Content-Range field with a byte-range-resp-spec of \"*\". The instance-length
1351 specifies the current length of the selected resource. A response with status code 206 (Partial Content)
1352 MUST NOT include a Content-Range field with a byte-range-resp-spec of \"*\".

1353 **7.5.1.15 417 ("Expectation Failed")**

1354 This response code is used to indicate that an expectation given in an Expect request-header field cannot
1355 be met by the server or that the server does not support the given expectation.

1356 [RFC 2616] states, "If a server receives a request containing an Expect field that includes an
1357 expectation-extension that it does not support, it MUST respond with a 417 (Expectation Failed) status."

1358 **7.5.1.16 500 ("Internal Server Error")**

1359 This response code is used to indicate that the server has an internal problem and is a generic server
1360 error response.

1361 **7.5.1.17 501 ("Not Implemented")**

1362 This response code is used when a client attempts to use a feature of HTTP (such as a method) that the
1363 server does not support.

1364 [RFC 2616] states, "The recipient of the entity MUST NOT ignore any Content - (e.g., Content-Range)
1365 headers that it does not understand or implement and MUST return a 501 (Not Implemented) response
1366 in such cases."

1367 **7.5.1.18 503 ("Service Unavailable")**

1368 This response code is used when a server, due to a temporary overload condition, is unable to service a
1369 request.

1370 **7.5.2 Minimal Understanding**

1371 Should a client wish to operate with minimal understanding of HTTP response codes, it need only
1372 examine the first digit of the response code to understand the general category of the response and "treat
1373 any unrecognized response as being equivalent to the x00 status code of that class, with the exception
1374 that an unrecognized response MUST NOT be cached." [RFC 2616].

1375 7.6 **Application Payload Syntax**

1376 Application payload message encoding using both XML [XML] and EXI [EXI] SHALL be supported
 1377 by all servers. Application payload message encoding using either XML [XML] or EXI [EXI] SHALL
 1378 be supported by all clients.

1379 7.6.1 **XML Encoding**

1380 The XML declaration is optional as per [XML] and SHOULD NOT be included in SEP 2 transactions,
 1381 to reduce packet sizes. The XML version used SHALL be 1.0. For XML payloads, the encoding
 1382 SHALL be UTF-8.

1383 7.6.2 **EXI Encoding**

1384 The options for encoding EXI documents SHALL be as follows, and transactions will likely fail if
 1385 different options are declared in the EXI option header. Options marked as (default) are EXI
 1386 specification [EXI] defaults and SHOULD NOT be specified explicitly.

- 1387 • Non-strict schema-informed grammar with the schema [ZB 13-0201]
- 1388 • Alignment is bit-packed (default)
- 1389 • Compression is false (default)
- 1390 • Strict is false (default)
- 1391 • Fragment is false (default)
- 1392 • Preserve options are all false (default)
- 1393 • selfContained is false (default)
- 1394 • schemaId is "S0" (two bytes: 0x53, 0x30, without quotes). This schemaId corresponds to the
 1395 normative schema of SEP 2.0 [ZB 13-0201].
- 1396 • datatypeRepresentationMap is not used (default)
- 1397 • valueMaxLength is unbounded (default)
- 1398 • valuePartitionCapacity is unbounded (default)
- 1399 • No user defined meta-data

1400 The following XML document describes the EXI option header that SHALL be used to encode the
 1401 messages.

```
1402 <header xmlns="http://www.w3.org/2009/exi">
1403   <common><schemaId>S0</schemaId></common>
1404 </header>
```

1405 7.7 **Content Negotiation**

1406 A client SHALL declare acceptable media types using the HTTP Accept header.

1407 7.7.1 **Schema Version Negotiation**

1408 When specifying an "application/sep-exi" media type, an extensibility level ("level") SHALL be
 1409 specified in the HTTP Accept header for schema version negotiation. "level" is an Accept-extension
 1410 defined in [RFC 2616]. The extensibility level specified using an HTTP Accept header supersedes an
 1411 extensibility declaration discovered during resource discovery (see Section 9.3).

1412 The Extensibility Level defines the base schema and its capability for arbitrary extension. The
 1413 Extensibility Level is one of "-S0" or "+S0". The S0 indicates the base schema version: SEP 2. "-S0"

1414 indicates the node does not accept arbitrary tags that are not defined in the base schema, and "+S0"
1415 indicates it accepts arbitrary tags. A node with "-S0" will likely fail on an EXI document using arbitrary
1416 types, elements, and attributes that are not defined in the schema used for encoding. Devices SHALL
1417 NOT send messages to nodes that declare "-S0" using arbitrary types, elements, and attributes.

1418 The grammar used for EXI SHALL be generated as a non-strict grammar only, as having both strict and
1419 non-strict grammars would put a large burden on storage requirements for certain devices. The use of a
1420 non-strict grammar allows for extensions without schema modification. An invalid (i.e., not defined in
1421 the schema) part of an EXI document is allowed in a non-strict grammar and can carry arbitrary tags,
1422 attributes, and text encoded using the built-in grammar.

1423 Due to strict memory constraints, some nodes may not be able to parse invalid parts of an EXI document
1424 encoded using the built-in grammar. To avoid such errors, a node may declare its inability to receive
1425 arbitrary extensions using the "-" (minus) prefix in the Extensibility Level. Alternatively, nodes that
1426 declare the "+" (plus) prefix in the Extensibility Level will be able to parse extended parts of an EXI
1427 document.

1428 Note that the Extensibility Level does not indicate whether the node can process the data, but only
1429 whether it can parse the data.

1430 The format of the Extensibility Level is "(-|+)Sn" where n is a character to describe base schema version
1431 (currently "0"). As extensions of SEP 2.x schemas are intended to be backward compatible, a node that
1432 declares schemaId "S[i]" is intended to be compatible with all versions between "S0" and "S[i]".

1433 For example:

1434 Accept: application/sep-exi; level=-S0

1435 indicates that the client wishes to receive content encoded using EXI where the base schema of the client
1436 is SEP 2 and that the client does not accept arbitrary tags not defined in the schema.

1437 A client SHOULD use the Extensibility Level discovered during resource discovery to determine if a
1438 server accepts non-strict parts of an EXI document prior to initiating PUT / POST operations where the
1439 content contains extended attributes / elements. A server SHOULD use the Extensibility Level specified
1440 in the Accept header to determine if a client accepts non-strict parts of an EXI document prior to
1441 responding to GET operations where the content contains extended attributes / elements.

1442 8 Security

1443 Depending on the underlying physical network, messages may be encrypted at lower layers, in addition
 1444 to the security features provided specifically for the application layer. This section describes the security
 1445 features that are provided at the application layer and that are REQUIRED for use over all networks.

1446 8.1 Introduction

1447 Securing transactions between clients and servers is based on using HTTP over TLS [RFC 2818] (also
 1448 known as HTTPS) using TLS version 1.2 [RFC 5246]. The TLS records are then transported using TCP.
 1449 The TLS handshake mechanism provides mutual authentication based on device certificates or self-
 1450 signed certificates and TLS records provide encryption and message authentication using the AES-CCM
 1451 mode of operation. Access control lists allow or deny use of resources based on authentication level and
 1452 address information. A registration list is used for authorizing clients.

1453 8.2 Security Attributes

1454 In this section we define some abstract data structures for managing registration (see Section 8.9) and
 1455 access control. How this functionality is accomplished is left to the implementer. No access to these data
 1456 structures is defined in this specification.

1457 8.2.1 Local Registration Attributes

1458 Local registration attributes represent the data which would be used to hold information passed out-of-
 1459 band as part of the registration process prior to resources being established.

1460 **Table 8-1 Local Registration Attributes.**

Attribute	Identifier	Type	Range	Description	Default
<i>aclLocalRegistration</i>	0x00	List	-	A table of RegistrationDescriptors each with information for a specific registration	(empty)
<i>aclLocalRegistrationEntries</i>	0x01	Integer	Implementation specific	Number of entries in <i>aclRegistration</i>	0

1461

1462 **Table 8-2 Local Registration Descriptor Entry.**

Name	Type	Range	Description	Default
PIN	Integer	0 – 999999	6-digit PIN (5 plus check digit) for basic server validation. The PIN is also reflected in the Registration resource linked to the EndDevice resource	-
SFDI	SFDI	-	SFDI of registering device. The SFDI is also reflected in the EndDevice resource	-
DeviceType	Integer	0 – 3	Minimum required device type	0
HardwareModuleName	String	-	Optional additional hardware module information	-

1463 8.2.2 **Access Control List (ACL) Attributes**

1464 Access control list (ACL) attributes represent the data that would be used to hold information to
 1465 determine whether access to a particular resource by a particular client is allowed or denied. An ACL
 1466 can enforce more granular access control based on various criteria (e.g., client identity). Conceptually,
 1467 an ACL exists for every single accessible resource, however in practice it is likely only certain resources
 1468 with more complex access policies would require a representation of all the data specified in this
 1469 section.

1470 **Table 8-3 ACL Attributes.**

Attribute	Identifier	Type	Range	Description	Default
<i>aclDefaultAccess</i>	0x02	Access Descriptor	-	Default access to resource	-
<i>aclSpecificID</i>	0x03	List	-	A list of SpecificIDDescriptors for each specific client access to resource	-
<i>aclSpecificIDEntries</i>	0x04	Integer	Implementation specific	Number of entries in <i>aclSpecificID</i>	0

1471

1472 **Table 8-4 AccessDescriptor Entry.**

Name	Type	Range	Description	Default
Method	Bitmap	0x0 – 0xf	Bitmap of which methods are supported: 0x1: GET 0x2: PUT 0x4: POST 0x8: DELETE 0x10: HEAD	0x0
AuthType	Integer	0x0 – 0x0f	Bitmap of which authentication types are allowed: 0x1: No authentication 0x2: User authentication 0x4: Self-signed Certificate 0x8: Device Certificate Remaining bits are reserved for authentication types not defined by this specification but by an additional security policy	0x0
DeviceType	Integer	0 – 3	Device type: 0: Unknown	0

1473

1474 **Table 8-5 SpecificIDDescriptor Entry.**

Name	Type	Range	Description	Default
Access	Access Descriptor	-	Access levels required for client	-
IPAddr	IPAddr	-	IP Address of client	-
Port	Integer	0x0000 – 0xffff	Port of client 0: Wildcard (any port)	0

1475 Access control list (ACL) attributes provide a mechanism for granting and revoking privileges to use
 1476 specified methods with a particular resource, applicable to all resources described in this specification.
 1477 Access control more granular than a resource is out of scope for this specification. The mechanisms for
 1478 authentication and the binding of that authentication to a specified identity (such as an IP address) are
 1479 specified in Sections 8.5 and 8.6.

1480 ACLs are used to set default privileges and grant additional privileges, not to deny privileges. Thus, if a
 1481 given client's request does not explicitly meet the required privilege in the associated ACL, then that
 1482 client does not have that privilege. The default configuration of an ACL for a given resource means that
 1483 resource is not accessible to clients. In practice, this state only exists ephemerally and all ACLs will be
 1484 initialized appropriately at startup according to the security policy and will be subsequently modified
 1485 according to registration and authentication.

1486 Subordinate resources created dynamically will initially inherit their ACL from their parent resource.

1487 ACLs may be statically fixed with a default operation for some resources and may be dynamic and
 1488 extensible for others. If a resource does not have an ACL, access is granted to the resource
 1489 unconditionally.

1490 Initialization of ACLs, beyond minimal requirements, is out of scope for this specification and is
 1491 governed by overall security policy. Default settings for ACLs for particular function sets are described
 1492 in Section 8.8.

1493 8.2.2.1 **aclDefaultAccess**

1494 *aclDefaultAccess* in the ACL is used for settings irrespective of the client identity of an incoming
 1495 request and is used initially to authorize an incoming HTTP request.

1496 8.2.2.2 **aclSpecificIDList**

1497 A SpecificIDDescriptor entry in the *aclSpecificIDList* part of the ACL is an additional entry used to
 1498 allow specific additional checks to be done to authorize based on client identity (IP address and port).

1499 8.2.2.3 **Access Authorization**

1500 These are controls that are independent of the source identity (IP address and port) and thus can be
 1501 configured in both *aclDefaultAccess* and a SpecificIDDescriptor entry in *aclSpecificIDList* as shown in
 1502 Table 8-4 and Table 8-5.

1503 In the following, 'corresponding ACL entry' means:

- 1504 • The first SpecificIDDescriptor entry in *aclSpecificIDList* which matches the incoming client
 1505 HTTP request's source IP address and port.
- 1506 • *aclDefaultAccess* otherwise

1507 8.2.2.3.1 **Method Attribute**

1508 The Method attribute is used to control which HTTP request method is allowed for client access.

- 1509 • GET
- 1510 • PUT
- 1511 • POST
- 1512 • DELETE

1513 The HTTP method of an incoming HTTP request is checked and Method authorization will be TRUE if
 1514 the method is allowed in the corresponding ACL entry.

1515 8.2.2.3.2 **AuthType Attribute**

1516 The AuthType attribute is used to control the required authentication types the client can use in its
 1517 incoming HTTP/HTTPS request.

- 1518 • 0x1: No authentication
- 1519 • 0x2: User authentication
- 1520 • 0x4: Self-signed Certificate
- 1521 • 0x8: Device Certificate

1522 If an incoming HTTP request is destined for the port associated with HTTPS (typically port 443), the
 1523 incoming authentication type will be set to the corresponding TLS session authentication type. If an
 1524 incoming HTTP request is destined for the port associated with HTTP (typically port 80), the
 1525 authentication type will be set to 0x1 (no authentication). The user authentication AuthType (0x02) can
 1526 be used if additional user authentication outside of the scope of this specification takes place.

1527 The incoming authentication type is compared with the bitmap in AuthType in the corresponding ACL
 1528 entry and AuthType authorization will be TRUE if the corresponding authentication type is set in
 1529 AuthType in the corresponding ACL entry. Representing this logically:

```
1530     if (authentication type & AuthType) != 0:  

  1531         AuthType authorization = TRUE  

  1532     else  

  1533         AuthType authorization = FALSE
```

1534 8.2.2.3.3 **DeviceType Attribute**

1535 The DeviceType attribute is used to control the device type required of the client's incoming HTTP
 1536 request. It is based on the deviceType OID in the certificate (see Section 8.11.7.1).

1537 If an incoming HTTP request is destined for the port associated with HTTPS, the incoming device type
 1538 will be set to the corresponding TLS session device type based on the deviceType OID in the certificate.
 1539 If an incoming HTTP request is destined for the port associated with HTTP, the device type will be set
 1540 to 0 (unknown).

1541 If the DeviceType in the ACL is set to 0, device type authorization will be TRUE unconditionally.
 1542 Otherwise the incoming device type is compared with the DeviceType in the corresponding ACL entry
 1543 and DeviceType authorization will be TRUE if the device type is equal to the DeviceType in the
 1544 corresponding ACL entry.

1545 8.2.2.4 **Authorization Logic**

1546 Authorization is granted if Method authorization, AuthType authorization, and DeviceType
 1547 authorization are all TRUE. Access to the resource can then take place.

1548 If Method authorization is not granted, the server MAY respond with either:

1549 HTTP/1.1 400 Bad Request

1550 Or
 1551 HTTP/1.1 405 Method Not Allowed
 1552 If AuthType or DeviceType authorization is not granted, the server SHOULD immediately respond:
 1553 HTTP/1.1 404 Not Found
 1554 The server MAY respond:
 1555 HTTP/1.1 401 Unauthorized
 1556 Otherwise, processing will continue.

1557 8.2.2.5 **ACL Examples**

1558 This section contains two informative examples to illustrate the use of ACLs, using EndDeviceList and
 1559 EndDevice.

1560 8.2.2.5.1 **EndDeviceList**

1561 The EndDeviceList resource is usually accessible by any client to find its own entry in the list.
 1562 Therefore, the ACL will be as follows:

1563 **Table 8-6 Example ACL for EndDeviceList.**

Attribute	Comment
<i>aclDefaultAccess</i>	Method: 0x01 (GET only) AuthType: 0x1 (no authentication) Device Type: 0 (unknown)
<i>aclSpecificID</i>	Empty
<i>aclSpecificIDEntries</i>	0

1564 In this example, there will never be any SpecificIDDescriptors required, as this resource needs to be
 1565 accessible to any device.

1566 8.2.2.5.2 **EndDevice**

1567 In this example, an EndDevice resource for a client does not exist prior to registration.

1568 The earliest point it can be created is at the point of registration and the ACL would be set as follows:

1569 **Table 8-7 Example ACL entry for EndDevice at point of registration.**

Attribute	Comment
<i>aclDefaultAccess</i>	Method: 0x00 (no default access) AuthType: 0x1 (no authentication) Device Type: 0 (unknown)
<i>aclSpecificID</i>	Empty
<i>aclSpecificIDEntries</i>	0

1570 In this example, there is no default access as only the client device associated with the EndDevice is able
 1571 to access the EndDevice. Also, at this point, there has been no client communication therefore there
 1572 would be no SpecificIDDescriptor entry in *aclSpecificIDList*.

1573 There would also be an entry placed in the local registration list corresponding to the client:

1574 **Table 8-8 Example local registration entry for registering device.**

Name	Type	Range	Description	Default
PIN	Integer	0 – 999999	6-digit PIN (5 plus check digit) of registering device	-
SFDI	SFDI	0 - 68719476735, 2^36-1	SFDI of registering device	-
DeviceType	Integer	0 – 3	Registering device type	0
HardwareModuleName	String	-	Optional additional hardware module information	-

1575 In this state, it is primed to be populated with a SpecificIDDescriptor when the device actually accesses
 1576 the EndDeviceList resource.

1577 When the client attempts access to any resource on a function set server using TLS, it will start
 1578 performing the TLS handshake, which involves transferring the client's certificate. At the point of
 1579 access, if there is a pending registration, it will be checked against the client's device. If there is a match,
 1580 and the validated certificate's SFDI matches, the ACL for the EndDevice will be populated with an
 1581 additional SpecificIDDescriptor:

1582 **Table 8-9 Example ACL entry for EndDevice at point of client access.**

Attribute	Comment
<i>aclDefaultAccess</i>	Method: 0x00 (no default access) AuthType: 0x1 (no authentication) Device Type: 0 (unknown)
<i>aclSpecificID</i>	One SpecificIDDescriptor for client
<i>aclSpecificIDEntries</i>	1

1583

1584 **Table 8-10 Example SpecificIDDescriptor for EndDevice at point of client access.**

Name	Type	Range	Description	Default
Access	Access Descriptor	-	Access levels required for client: Method: 0xF (GET, PUT, POST, DELETE) AuthType: 0x8 (Device Certificate) DeviceType: (as appropriate for device)	-
IPAddr	IPAddr	-	IP Address of client	-
Port	Integer	0x0000 – 0xffff	Port of client	0

1585

8.3 Device Credentials

1586 There are three credentials per device:

- 1587 • Short Form Device Identifier (SFDI)

- 1588 • Long Form Device Identifier (LFDI)
1589 • PIN

1590 8.3.1 **Certificate Fingerprint**

1591 The certificate fingerprint is the result of performing a SHA256 operation over the whole DER-encoded
1592 certificate and is used to derive the SFDI and LFDI. A certificate fingerprint is not confidential and is
1593 never used to derive subsequent keying material.

1594 An example certificate fingerprint used for illustration in the following examples is:

1595 3E4F-45AB-31ED-FE5B-67E3-43E5-E456-2E31-984E-23E5-349E-2AD7-4567-2ED1-45EE-213A

1596 8.3.2 **Short-form Device Identifier (SFDI)**

1597 The SFDI SHALL be the certificate fingerprint left-truncated to 36-bits. For display purposes, this
1598 SHALL be expressed as 11 decimal (base 10) digits, with an additional sum-of-digits checksum digit
1599 right-concatenated. Based on the example in Section 8.3.1, this would be 167-261-211-391.

1600 Left truncation to 36-bits: 0x3E4F45AB3

1601 Expressed as a decimal: 16726121139

1602 Right-concatenation of check digit and hyphenation: 167-261-211-391

1603 For input validation purposes, the sum of the digits of the fingerprint including the checksum digit,
1604 modulo 10, SHALL be zero. The SFDI has sufficient entropy (2^{36}) to uniquely identify the device in the
1605 context of its usage and is used to identify a device within a HAN or site domain. It should not be used
1606 in a truly global context (i.e., where the identity of the device cannot be qualified with the domain it is
1607 in).

1608 For a device with a Device Certificate, the SFDI can be printed on the device packaging.

1609 8.3.3 **Long-form Device Identifier (LFDI)**

1610 The LFDI SHALL be the certificate fingerprint left-truncated to 160-bits (20 octets). For display
1611 purposes, this SHALL be expressed as 40 hexadecimal (base 16) digits in groups of four. Based on the
1612 example in Section 8.3.1, this would be '3E4F-45AB-31ED-FE5B-67E3-43E5-E456-2E31-984E-23E5'.
1613 The LFDI is used when a globally unique identity is required, for example in sending an event back to a
1614 service provider that is associated with a particular device.

1615 8.3.4 **6-digit PIN code**

1616 The SFDI and LFDI are derived from public information (i.e., a Certificate), therefore can potentially be
1617 recreated by an eavesdropper. Therefore, a device MAY also have an additional 6-digit PIN code, which
1618 can be shared out-of-band with a service provider in conjunction with the SFDI or LFDI. For display
1619 purposes, this SHALL be expressed as 5 decimal (base 10) digits, with an additional sum-of-digits
1620 checksum digit right-concatenated:

1621 Original PIN: 12345

1622 Right-concatenation of check digit and hyphenation: 123-455

1623 For input validation purposes, the sum of the digits of the PIN including the checksum digit, modulo 10,
1624 SHALL be zero. The PIN MAY be obtainable from the EndDevice server through the Registration
1625 resource to validate that the client is in communication with the correct server. The PIN SHOULD be
1626 configurable on a device where possible for registration purposes, otherwise SHOULD be a random 5-
1627 digit value plus check digit pre-programmed into the device and printed on the device. The PIN is not
1628 overly secure and therefore SHALL NOT be used in any way to derive keys for actual data encryption.

1629 8.3.5 **Registration Code**

1630 The SFDI and PIN are usually presented separately. However, in certain cases it may be convenient to
1631 provide a single registration code, which is simply the concatenation of the SFDI and the PIN expressed
1632 as a decimal (base 10) number:

1633 SFDI || PIN

1634 From the examples above, this would be 167-261-211-391-123-455.

1635 8.4 **Resource Access Authentication and Authorization context**

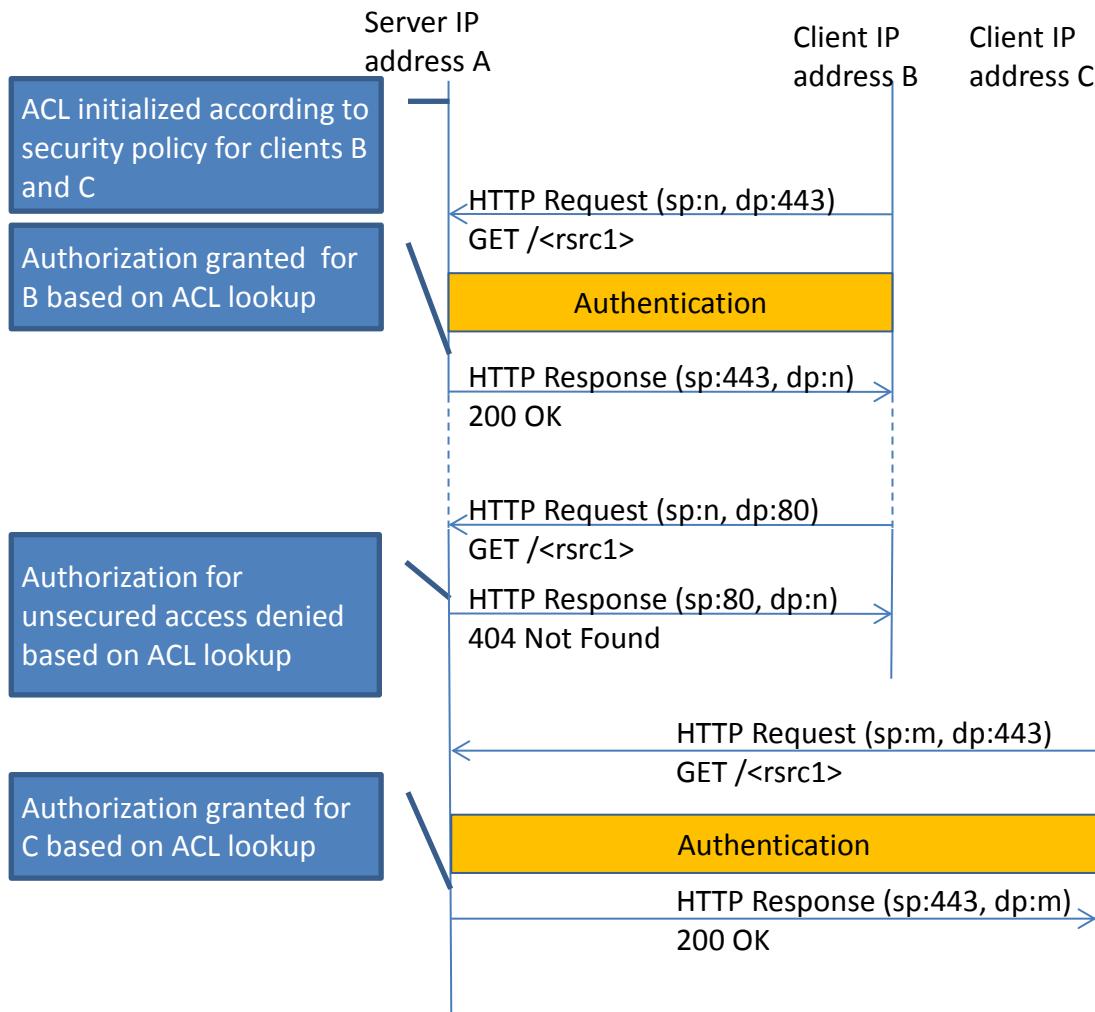
1636 A node is able to perform network layer communication once it has been authenticated and authorized to
1637 join the network. However, application layer authentication and authorization MAY be required before
1638 clients and servers can exchange application layer messaging. Registration (see Section 8.9) with a
1639 utility or third party service provider MAY also be needed to provide explicit device and user
1640 authorization at the application layer.

1641 Resource access requiring application layer authentication, data confidentiality and integrity checking
1642 SHALL occur through requests from a client to the server using HTTP over TLS [RFC 2818] (also
1643 known as HTTPS) using TLS version 1.2 [RFC 5246]. Resource access not requiring application layer
1644 authentication, data confidentiality or integrity checking SHALL occur through requests from a client to
1645 the host server using HTTP [RFC 2616].

1646 If a request is made to the port associated with HTTPS, it is considered an HTTPS request and
1647 authentication SHALL have taken place. If authentication has not taken place, authentication SHALL be
1648 initiated as described in Section 8.5. When authenticated, the request is then passed to the ACL
1649 associated with the resource. Ancillary information about the request obtained from the secure session,
1650 notably the level of client authentication, will also be compared with the ACL.

1651 If the request is made to the port associated with HTTP, it is considered an HTTP request and
1652 authentication SHALL NOT be REQUIRED and the request is then passed to the ACL associated with
1653 the resource. Ancillary information about the request stating the client is unauthenticated will also be
1654 compared with the ACL (see Section 8.2.2.3.2).

1655 Authorization on a request-by-request basis is determined by the ACL settings for the resource, which
1656 may be set up at the end of the authentication based on the level of client authentication. The Local
1657 Registration List (*aclLocalRegistrationList*) may be additionally used to authorize on a device-by-device
1658 basis.



1659

1660

Figure 8-1: Example Device Authentication Procedure Using HTTPS Port 443

1661 8.5 Resource Access Authentication

1662 Resource access authentication only applies using HTTPS. It may be possible to authenticate at a higher
1663 level using authentication based on HTTP-only transactions but this is out of scope for this specification.

1664 The use of TLS [RFC 5246] requires that all hosts implementing server functionality SHALL use a
1665 Device Certificate whereby the server presents its Device Certificate as part of the TLS handshake.

1666 The application authentication process is as follows:

- 1667 1) The resource's server listens on the TCP port associated with HTTPS.
- 1668 2) The client initiates an HTTP request using a random unused source TCP port to the resource's
1669 server using the TCP port associated with HTTPS.
- 1670 3) If no TLS session is in place, a TLS handshake SHALL occur between the client and server:
 - a) Authentication of the server SHALL be done as part of the TLS handshake by validating its Device Certificate as described in [RFC 5246], Section 7 using the inherent PKI. If security policy dictates, additional certificate validation MAY be required.

- 1675 b) If the client has a Device Certificate, authentication of the client SHALL be done as part
 1676 of the TLS handshake by validating the client's Device Certificate as described in [RFC
 1677 5246], Section 7 using the inherent PKI. If security policy dictates, additional certificate
 1678 validation MAY be required. The authentication level to be compared with a resource's
 1679 corresponding AuthType attribute will be 0x8 (Device Certificate).
 1680 c) If the client has a Self-signed Certificate, the Self-signed Certificate SHALL be
 1681 validated for correctness. The authentication level to be compared with a resource's
 1682 corresponding AuthType attribute will be 0x4 (Self-signed Certificate).

1683 If the client does not have a certificate and the security policy allows, client authentication MAY NOT
 1684 need to take place, or secondary client authentication MAY take place after the TLS handshake. If
 1685 secondary client authentication has taken place, the authentication level to be compared with a resource's
 1686 corresponding AuthType attribute will be 0x2 (User Authentication). If no client authentication has
 1687 taken place, the authentication level to be compared with a resource's corresponding AuthType attribute
 1688 will be 0x1 (No authentication).

1689 8.6 Resource Access Authorization

1690 Pre-authorization for resources is normally set when the client registers with the host as described in
 1691 Section 8.9. If the security policy allows, authorization MAY occur immediately after authentication
 1692 based on implicit rules to allow a request to complete. This is to allow unregistered access to resources
 1693 based on security policy. If the client uses a Self-signed Certificate, pre-authorization using the SFDI of
 1694 the Self-signed Certificate MUST have taken place and authorization SHALL be granted if the SFDI of
 1695 the presented Self-signed Certificate matches the SFDI presented as part of registration.

1696 8.7 Cipher Suites

1697 All devices SHALL support the TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite
 1698 [I-D AESCCM]. The ECC cipher suite SHALL use elliptic curve secp256r1.

- 1699 • All devices acting as a server SHALL support the ECC cipher suite. In particular, all devices
 1700 acting as a server SHALL have an ECC certificate.
- 1701 • All devices acting as a client SHALL support the ECC cipher suite for the purposes of
 1702 validating an ECC certificate.
- 1703 • All devices acting as a client SHOULD support the request for an ECC certificate.

1704 8.8 Default Security Policy

1705 Service providers create security policies by balancing the requirements of their regulatory environment
 1706 and the results of their risk assessments. Different regulatory environments may mandate requirements
 1707 that trade ease of data access with information assurance. The use of TLS, ACLs, and other security
 1708 controls give the service provider the flexibility to meet these needs. Security policies are a combination
 1709 of ACL attribute values and additional security controls dictated by the service provider.

1710 Implementation of security policies is out of scope of this specification. For the purpose of certification
 1711 testing, the following table represents the default security policy for each function set. Servers SHALL
 1712 be configurable to support each default policy for all implemented function sets during certification
 1713 testing.

1714 The Function Set column in Table 8-11 reflects the functionsImplemented attribute in
 1715 DeviceInformation.

1716

Table 8-11 Attribute values for Default Security Policy.

Function Set	<i>aclDefaultAccess</i> AuthType	Device Certificate needed	Registered Device
Device Capability	0xf	No	No
Self Device Resource	0xc	No	Yes
End Device Resource	0xc	No	Yes
Function Set Assignments	0x8	Yes	Yes
Subscription / notification mechanism	0x8	Yes	Yes
Response	0x8	Yes	Yes
Time	0x8	Yes	Yes
Device Information	0x8	Yes	Yes
Power Status	0x8	Yes	Yes
Network Status	0x8	Yes	Yes
Log / Event Log	0x8	Yes	Yes
Configuration Resource	0x8	Yes	Yes
Software Download	0x8	Yes	Yes
DRLC	0x8	Yes	Yes
Metering	0x8	Yes	Yes
Pricing	0xc	No	Yes
Messaging	0xc	No	Yes
Billing	0x8	Yes	Yes
Prepayment	0x8	Yes	Yes
Flow Reservation	0x8	Yes	Yes
DER Control	0x8	Yes	Yes

1717

1718 The *aclDefaultAccess* attribute Method value SHOULD match the Allowed Methods for each resource
 1719 enumerated in the SEP 2 WADL [ZB 13-0201]. The Method value MUST contain GET (0x01). The
 1720 *aclDefaultAccess* attribute DeviceType value should be 'unknown' (0). Servers SHALL support the
 1721 default policies for certification testing. Servers MAY additionally support alternative policies. For

1722 example, to meet regulatory requirements a utility may mandate a policy that provides unauthenticated
1723 pricing information from a pricing server over the port associated with HTTP to any SEP 2 device.
1724 Based on risk assessments, service providers may have differing policies for devices enrolled in high-
1725 incentive Demand Response / Load Control programs than those enrolled in low-incentive programs, to
1726 include additional requirements such as DeviceType authorization. Servers SHOULD provide the
1727 functionality to support multiple security policies to meet the requirements of different service
1728 providers.

1729 **8.9 Registration**

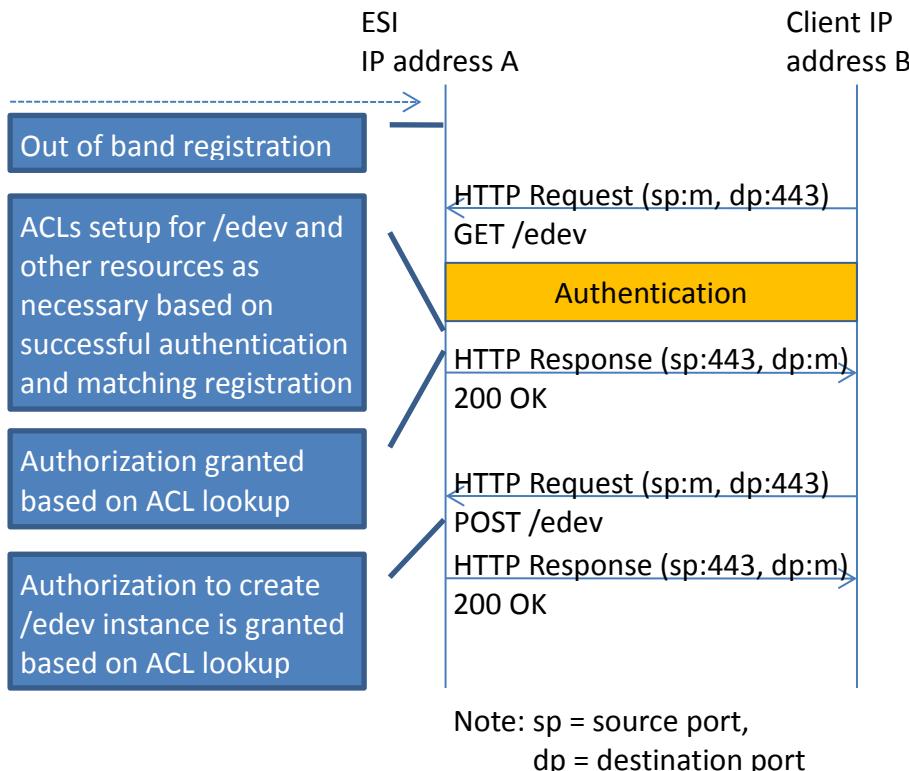
1730 Registration describes the procedure whereby an out-of-band procedure is used to convey client
1731 registration information a priori to the server that houses a resource that will subsequently be accessed
1732 by the client. The registration information is the client's SFDI and optionally, PIN, which uniquely
1733 identifies the client in the given context.

1734 Registration may occur some time before the client attempts to access a resource, for example, using a
1735 web site or telephone to register the information with a service provider. The service provider will then
1736 provide the information to the EndDevice server using some out-of-band mechanism, (e.g., the AMI
1737 network) and the server will program its registration list accordingly.

1738 Alternatively, there may be no actual registration before a client attempts to access a resource and, for
1739 example, the server may present the premises owner with the SFDI of the client attempting access via a
1740 user interface. The premises owner may then continue to authorize the client access, or deny access
1741 based on the information presented from the client's certificate.

1742 This section describes a typical registration procedure for a client using a Device Certificate with an
1743 EndDevice server.

1744 Registration for clients SHALL occur via an EndDevice resource corresponding to the client, which
1745 typically resides on an ESI associated with the utility, premises owner or third party service provider
1746 that is trusted to perform registration.



1747

1748 **Figure 8-2: Device authentication with registration procedure examples using HTTPS**1749 8.9.1 **EndDeviceList**

1750 Clients SHALL locate HAN services by performing DNS Service Discovery (DNS-SD) queries to the
 1751 HAN, see Section 9 for details. The client can then resolve the URI of the EndDeviceList (given as
 1752 /edev for illustration purposes) for registration and authentication purposes and know which port(s) the
 1753 server for the EndDeviceList is listening on.

1754 The EndDeviceList is the resource used by a client to complete the process initiated by registration of
 1755 the client when the device owner wishes to register the device in a utility, premises owner or service
 1756 provider program. In some cases, registration MAY be required for access.

1757 Upon registering a client, the EndDevice resource's server *aclLocalRegistrationList* will be configured
 1758 with:

- 1759 • The client SFDI, to be registered in the utility, premises owner or third party service
 1760 provider program.
- 1761 • Optionally, the client PIN
- 1762 • The required device types of the associated client.

1763 Thus, at the point of registration, the EndDevice resource's server is able to perform authentication based
 1764 on the Device Certificate and additional user authentication based on the client SFDI.

1765 The EndDevice resource's server MAY allow access from clients that have not been pre-configured if
 1766 the security policy allows.

1767 The registration procedure is as follows:

- 1768 1. The EndDevice resource's server SHALL listen on the TCP port associated with HTTPS and
 1769 follow the procedure described in [RFC 5246] when a client attempts to access the EndDevice
 1770 resource.
- 1771 2. Authorization SHALL then occur whereby the ACLs of the server resources corresponding to
 1772 the registering client are set according to the security policy and the presence in
 1773 *aclLocalRegistrationList*. This ensures that following registration, a client can typically proceed
 1774 to access all the resources it is authorized to without having to perform any further procedures.
- 1775 3. If present, the client MUST verify that the EndDevice's associated Registration resource
 1776 contains the correct PIN for the client. If the PIN does not match, the client SHOULD NOT
 1777 attempt any further access to that server.
- 1778 4. The client SHALL subsequently re-use its Device Certificate to authenticate with any other
 1779 server host. It does not need to re-authenticate with the EndDevice resource's server.

1780 8.10 Security LogEvents

1781 There are specific LogEvents attributable to security. These will use a function set enumeration of
 1782 'security' as defined in the model. These are as follows:

1783 **Table 8-12 Security LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
SEC_TLS_ALERT	0x00	SHOULD be issued when a TLS Alert is generated. The logEventID SHALL be set to the TLS Alert value [RFC 5246].
SEC_REGISTRATION_MISS	0x01	SHOULD be issued when a received certificate does not have a corresponding SFDI entry in the registration list.
SEC_ACL_ACCESS_FAILED	0x02	SHOULD be issued when access to a resource fails due to failing access control criteria described in Section 8.2.2.

1784 8.11 Certificate Management

1785 8.11.1 Introduction

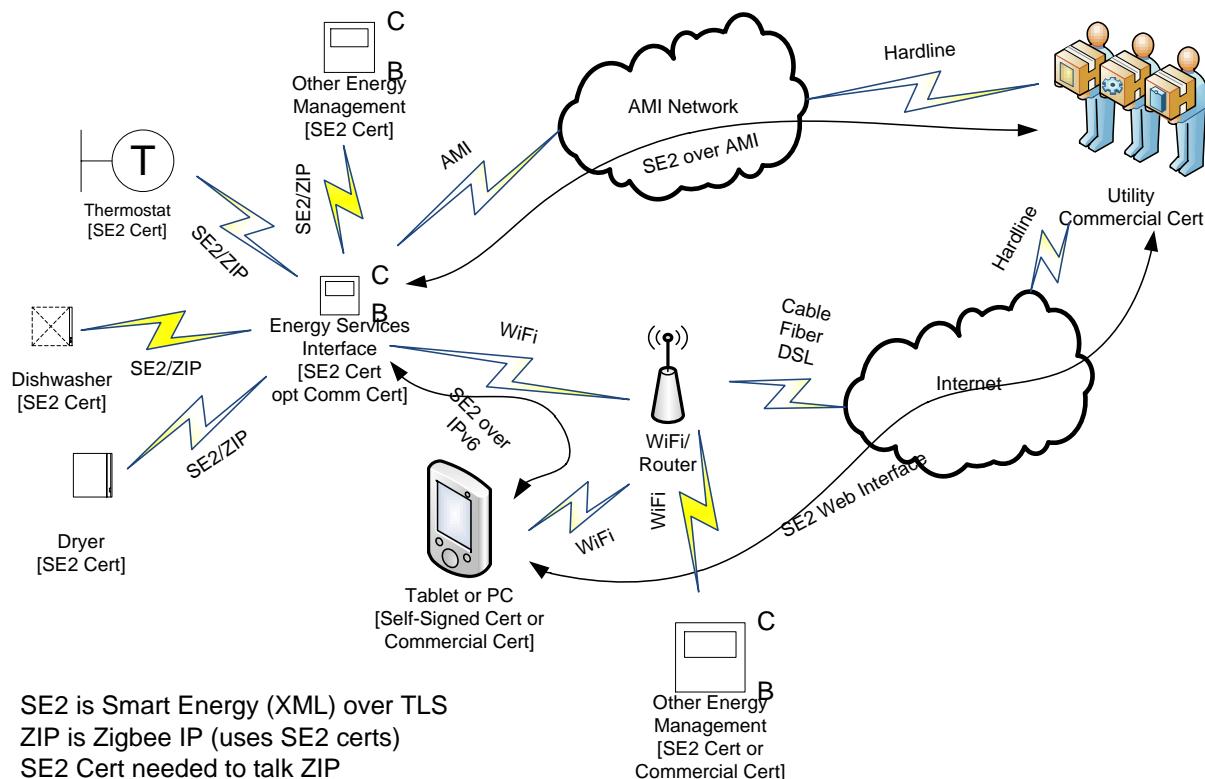
1786 There is a single agreed-upon Public Key Infrastructure (PKI) in the Smart Energy Profile 2.0 (SEP 2)
 1787 certificate management system: the Manufacturing PKI. The Manufacturing PKI issues certificates to
 1788 devices at the time of application installation, e.g., at manufacture. These certificates are intended for
 1789 use during deployment (or redeployment) and on-going operation to authenticate the device to other
 1790 SEP 2 devices implementing SEP 2 applications over TLS.

1791 In addition, this document describes a few other types of certificates that Smart Energy applications may
 1792 encounter depending on which services they support – see Section 8.11.8.3 below. While native SEP 2
 1793 devices are required to understand, support and process Manufacturing PKI certificates, support of the
 1794 additional certificates is optional and generally targeted at specific classes of devices (e.g., Energy
 1795 Services Interface (ESI), web portal).

1796 There are 6 classes of certificates that may be active in a SEP 2 system, depending on configuration and
 1797 use:

- 1798 • Device Certificates – Issued under the Manufacturing PKI during manufacturing to purpose-built (aka "native") SEP 2 certified devices for operational purposes.
- 1800 • Device Test Certificates – Issued under the Manufacturing PKI during manufacturing to purpose-built (aka "native") SEP 2 certified devices for test purposes.
- 1802 • Additional Certificates for SEP 2 Devices – One or more OPTIONAL TLS server certificates issued by non-SEP 2 CAs to SEP 2 devices such as ESI's for use in addition to the Device Certificate.
- 1803 • Generic Client Certificate for non-native entities – A TLS client certificate issued by a generic (non-SEP 2) Certificate Authority to a non-native entity.
- 1805 • Generic Server Certificate for non-native entities – A TLS server certificate issued by a generic (non-SEP 2) Certificate Authority to a non-native entity.
- 1807 • Self-Signed Client Certificate for non-native entities – A TLS client certificate self-generated and self-signed by a customer or software.

1809 A diagram of a deployed solution based on SEP 2 devices is illustrated in Figure 8-3.
 1811



1813 **Figure 8-3: SEP 2 including ZigBee IP deployment**

1814 With the exception of the "AMI Network", the "Internet" and the "Utility", all of the devices pictured
 1815 above reside in the customer premises. Devices purpose-built for SEP 2 have a manufactured-in Device
 1816 Certificate. Other devices and entities such as the utilities and a customer-owned tablet computer may

1817 use self-signed certificates or may use certificates issued by commercial CAs. Support for those
 1818 certificates is mainly dependent on the options implemented by the server.

1819 **8.11.2 Certificate Usage – Authentication vs. Authorization**

1820 Certificates provide a mechanism to authenticate an identity. Once authenticated (by proving possession
 1821 of the associated private key, and by having the certificate chain to a known root of trust), that identity
 1822 (or the service, person, or application associated with that identity) can be authorized access to
 1823 resources, the ability to assume a role (e.g., system operator, general user) or perform various functions.

1824 An authenticated certificate by itself does not generally grant authorization. Specific applications that
 1825 accept the certificate MAY grant implicit authorization to access any resource under the purview of the
 1826 application, but usually authorize access based on the identity represented by the certificate (e.g., an
 1827 Access Control List entry). The following tables describe both the authentication and authorization uses
 1828 of certificates described by this profile.

1829 Table 8-13 describes the mechanisms for certificate validation – the authentication of the identity
 1830 claimed in the certificate. Note that for a self-signed certificate, the authentication is limited only to
 1831 validating the self-signature over the certificate and that provides only an integrity check.

1832 The phrase "SEP 2 Cert – Indef" is meant to convey that Manufacturing PKI certificates are indefinitely
 1833 valid and the check is limited solely to a check of the signatures on the certificate chain. The phrase
 1834 "Optional OCSP" means that the server device (and optionally, the client device) may utilize Online
 1835 Certificate Status Protocol (OCSP) [RFC 2560] as an additional mechanism to determine if a certificate
 1836 has been revoked. OCSP may only be used to verify non-SEP 2 certificates.

1837 Table 8-14 describes the mechanisms for determining if the holder of a specific certificate (and its
 1838 private key) may access specific resources. Generically, each resource has an ACL tagged to it. If the
 1839 identity of the certificate holder has the appropriate rights to a specific resource, then the representation
 1840 of the resource is returned via query. Note that there are certain resources that are accessible to all
 1841 authenticated clients.

1842 **Table 8-13 TLS Authentication Matrix.**

Authentication		Server		
		Native SEP 2 Application	Generic Server	Self-Signed
Client	Native SEP 2 Application	SEP 2 Cert - Indef	Optional OCSP	Not Allowed
	Generic Client	Optional OCSP	Not Specified Here	Not Specified Here
	Self-Signed	Signature Validation	Not Specified Here	Not Specified Here

1843
 1844 The above table should be read as describing the authentication on the offered client certificate by the
 1845 server. For example – Generic Client / Native SEP 2 Application has an "Optional OCSP" entry
 1846 meaning the SEP 2 Application can OPTIONALY use OCSP to check the validity of the Generic
 1847 Client certificate in addition to its normal certificate validation process.

1848 "SEP 2 Cert – Indef" means that if the Smart Energy Manufacturing PKI certificate validly chains to the
 1849 Smart Energy root it is considered valid – neither OCSP nor CRLs are used or issued by CAs within the
 1850 Manufacturing PKI hierarchy.

1851 **Table 8-14 SEP 2 Authorization Matrix.**

Authorization	Server		

	Native SEP 2 Application	Generic Server	Self-Signed
Native SEP 2 Application	ACL	ACL or Public Resources (Server specific)	Not Allowed
Generic Client	ACL or Public Resources	Not Specified Here	Not Specified Here
Self-Signed	ACL or Public Resources	Not Specified Here	Not Specified Here

1852

1853 The above table should be read as describing the authorization mechanism used by the server with
 1854 respect to the offered client credential. For example – the Generic Client/Native SEP 2 Application entry
 1855 of "ACL" means that either there is a specific ACL for the generic client credential that permits access
 1856 to specific data OR the absence of such ACL allows that generic client access only to public resources.

1857 8.11.3 Manufacturing PKI

1858 This section covers only those certificates issued under the auspices of the Manufacturing PKI. It
 1859 specifically excludes the certificates described in Section 8.11.8.3 as "Other Certificates".

1860 The SEP 2 Manufacturing PKI SHALL be a hierarchy with a depth of 2, 3 or 4 levels. At the top level,
 1861 Manufacturing PKI hierarchy SHALL have one SERCA. One or more SERCAs SHALL be nominated
 1862 by the CSEP [CSEP] and SHALL be out-sourced to a commercial CA service provider. A SERCA is the
 1863 property of the CSEP (specifically the private keys associated with the SERCA root are owned by
 1864 CSEP) and is operated on the CSEP's behalf by a commercial CA. A commercial CA MAY operate one
 1865 or more SERCAs as business needs dictate and this is one possible model for the initial operation of the
 1866 SERCAs. A SERCA MAY be moved from one commercial CA to another as business needs dictate.
 1867 Private key material for a SERCA MUST be held in a form to allow secure transfer of the material to a
 1868 new commercial CA if necessary. The SERCA issues MCA certificates and MICA certificates to
 1869 authorized vendors based on CSEP provided policy. The current version of the specification allows a
 1870 SEP 2 vendor to contract with a SERCA for the issuance of Device Certificates, but the actual
 1871 permission to issue such certificates rests with the CSEP.

1872 A SERCA MAY issue Device Certificates on behalf of one or more manufacturers.

1873 The Manufacturing PKI hierarchy MAY include one SEP 2 MCA. One or more MCAs SHALL be out-
 1874 sourced to a SEP 2 vendor, specifically for the issuance of vendor-specific MICAs. A SEP 2 vendor
 1875 MAY contract with a commercial CA to host the SEP 2 vendor's MCA credentials (certificate and
 1876 private key), but retains ownership of such credentials. MCAs may only issue MICA certificates.

1877 The Manufacturing PKI hierarchy MAY include one SEP 2 MICA. One or more MICAs SHALL be
 1878 out-sourced to a SEP 2 vendor, specifically for the production of SEP 2 certified devices by that vendor.

1879 All devices SHALL store exactly one SERCA in their certificate path. All devices SHALL include at
 1880 least the public keys of all existing SERCAs and MAY include their certificates. In the course of any
 1881 particular authentication, any device can therefore verify the chain of signatures leading up to any one of
 1882 the roots.

1883 The following certificate paths are the only valid instantiations of the Manufacturing PKI:

- 1884 • SERCA -> Device Certificate
- 1885 • SERCA -> MICA -> Device Certificate

- 1886 • SERCA -> MCA -> MICA -> Device Certificate

1887 **8.11.3.1 Manufacturing Certificate Lifecycle**

1888 Certificates within the Manufacturing PKI have an indefinite lifetime – this includes, specifically, a
 1889 SERCA. Nevertheless, CA certificates may be retired and subsequently replaced when circumstances
 1890 dictate. In such cases, the retired certificate and its associated private key SHALL no longer be used for
 1891 issuing certificates. However, parties MAY continue to rely upon those certificates for validating
 1892 subordinate certificates. If, however, the signature algorithm or parameter set used in a manufacturing
 1893 certificate comes to be viewed as insufficiently secure for the purpose, parties MAY retire those
 1894 certificates and associated public keys. A retired certificate and key may no longer be used for issuing
 1895 new certificates, but is not considered revoked for the purpose of validation.

1896 An MCA or MICA certificate SHALL NOT be re-issued (e.g., re-signed with the same SubjectName
 1897 and public key, but with a new validity period). Instead, if it is desired to retire an existing
 1898 key/certificate, a new key pair SHALL be generated and bound into a new MCA or MICA certificate
 1899 generally with a new serialNumber component of the SubjectName. Operationally, the responsible CA
 1900 SHOULD verifiably¹ destroy the private key of the retired certificate. Retired certificates MUST remain
 1901 available to verify subsidiary certificates. A replacement of a MCA or MICA SHOULD use a new name
 1902 formed by incrementing or otherwise adjusting the serialNumber component of the subject name. See
 1903 Sections 8.11.8.2.1 and 8.11.8.2.2 for the format of the name.

1904 A side effect of the indefinite lifetime requirement coupled with the permanent embedding of the Device
 1905 Certificate and its certificate chain within a device is that the device, MCA and MICA certificates
 1906 MUST NOT and cannot be revoked once issued. Specifically, no MCA, MICA or SERCA shall issue or
 1907 be required to issue CRLs and no MCA or MICA shall operate or have operated on their behalf any
 1908 OCSP server for the purpose of providing validity information for any certificate under the
 1909 Manufacturing PKI hierarchy. This does not preclude the use of a certificate and its corresponding
 1910 identity to be used in a blacklist or whitelist for authorization purposes to allow or deny access to
 1911 resources on a server.

1912 **8.11.3.2 Device Certificate Lifetime**

1913 [NIST SP800-57] treats the question of the expected lifetime of various algorithm choices and key
 1914 lengths. For the purpose of this document, the Manufacturing PKI certificate uses a choice for algorithm
 1915 and key length that currently have no end-use date.

1916 **8.11.3.3 Device Certificate Validity**

1917 A Device Certificate issued by the Manufacturing PKI is considered valid indefinitely. However, the
 1918 validity of the certificate does not imply any authorizations for the holder of the certificate. Any relying
 1919 party is responsible for maintaining a mechanism for determining whether a given certificate is usable
 1920 (e.g., valid authenticator for specific resource, implicit authorization) in a given circumstance (e.g., an
 1921 access control list or other white or black list).

1922 **8.11.4 General Certificate Format**

1923 **8.11.4.1 RFC 5280 Compliance**

1924 All certificates in the Manufacturing PKI SHALL be compliant with [RFC 5280].

1925 **8.11.4.2 IEEE 802.1AR Compliance**

1926 Device Certificates and Device Test Certificates in the Manufacturing PKI SHALL be compliant with
 1927 [IEEE 802.1AR] with the following exceptions:

¹ The process for destroying private keys is a business issue that should be covered by any contract for SERCA services.

- 1928 • Device Certificates and Device Test Certificates take the general form of an iDevID certificate
1929 as defined in [IEEE 802.1AR].
- 1930 • Differing from [IEEE 802.1AR] requirements, the SubjectName field of the Device Certificate
1931 is empty as the X500 name form is not well suited to describe or identify physical serialized
1932 devices². The device identity is contained in the SubjectAlternativeName extension which
1933 contains a single GeneralName of type OtherName that is further sub-typed as a
1934 HardwareModuleName (id-on-HardwareModuleName) as defined in [RFC 4108]. Per
1935 [RFC 5280], this extension MUST be marked critical when the SubjectName field is empty. The
1936 hwType field of HardwareModule name is assigned by the SEP 2 manufacturer and SHOULD
1937 be different for each different type of manufactured device.

1938 8.11.5 General Restrictions and Conditions

1939 In addition, SEP 2 certificates have the following restrictions:

- 1940 • All SEP 2 certificates are X.509 v3 certificates as defined in [RFC 5280].
- 1941 • The only permitted public key type for SEP 2 certificates in the Manufacturing PKI is an Elliptic
1942 Curve (EC) public key on the NIST P-256 curve. (Note: See Sections 8.11.8.3.3 and 8.11.8.3.4
1943 for details on the use of RSA Public Keys and certificates.)
- 1944 • The signature method for signatures formed by EC P-256 private keys MUST be SHA256 with
1945 ECDSA.
- 1946 • Within the Manufacturing PKI hierarchy, all certificates MUST contain only an EC P-256
1947 public key. That public key MUST contain an elliptic curve point in uncompressed form. See
1948 [RFC 5480] Section 2.2 for details on the uncompressed form.
- 1949 • Per [RFC 5280], CAs MUST ensure the uniqueness of the serial numbers on the certificates
1950 they issue. CAs MAY use one of three mechanisms to meet this requirement: comparison
1951 against previously issued certificates, monotonically increasing serial numbers or random octet
1952 string. For the latter method, true random strings of 8 octets are sufficient for certificates issued
1953 by SERCAs or by MCAs, random strings of 10 octets are sufficient for certificates issued by
1954 MICAs that are planning to issue less than 50 million certificates, and 11 octets are sufficient for
1955 MICAs that are planning to issue less than 250 million certificates³.
- 1956 • Per [RFC 5280], the IssuerName of any certificate MUST be identical to the signer's
1957 SubjectName.
- 1958 • With the exception of Device Certificates and Device Test Certificates as described in Sections
1959 8.11.2.3 and 8.11.2.4, the SubjectName MUST be non-empty.

1960 8.11.6 Extensions

- 1961 • The certificatePolicy extension in any certificate consists of one or more PolicyInformation
1962 objects containing only the policyIdentifier field. The PolicyInformation object MUST NOT
1963 contain any policyQualifier fields. If present, the policyQualifier field SHOULD be ignored.

² Or rather there are too many different possible ways to represent these types of entities using X500 Distinguished Names. Rather than attempt to resolve the differences between each individual company's interpretation of SubjectName guidance in [IEEE 802.1AR], we constrain the identity expression to just the SubjectAlternativeName format described in [IEEE 802.1AR].

³ This is derived from the Birthday Collision problem where we want to set the chance of collision in the random space at less than 10^8 . 8 octets $\approx 600K$ certs, 10 octets $\approx 155M$ certs, 11 octets $\approx 2.5B$ certs signed without serial number collision.

- In the Manufacturing PKI hierarchy, each Device Certificate and the CAs that make up the Device Certificate's path contain one or more device type identifiers encoded in the certificatePolicy extension in the PolicyInformation: policyIdentifier field. These policyIdentifier Object Identifier (OID)s [RFC 5280] are taken from those OIDs defined under the deviceType arc of the [CSEP] OID tree. See Section 8.11.7.1 below for acceptable values and for information on the management of that arc.
- Each certificate in the Manufacturing PKI hierarchy MUST have a Valid: notBefore field consisting of the time of issue encoded as per [RFC 5280] Section 4.1.2.5 and a Valid:notAfter field consisting of the GeneralizedTime value `99991231235959Z` (see [RFC 5280], Section 4.1.2.5) for 256-bit ECC-based certificates.
- Each CA certificate MUST contain a SubjectKeyIdentifier extension with an 8-octet key identifier generated as per method (2) of Section 4.2.1.2 of [RFC 5280]. A non-CA certificate MAY contain a SubjectKeyIdentifier extension – if it does, such extension MUST be generated as per method 2 of Section 4.2.1.2 of [RFC 5280]. In both cases, the extension MUST be marked non-critical.
- SEP 2 devices MUST be able to follow a chain where the key identifier was not generated in compliance with this section but where there is correspondence in actual values between a child AuthorityKeyIdentifier and a parent's (CA's) SubjectKeyIdentifier.
- Each certificate, except self-signed client certificates and root certificates, MUST contain an AuthorityKeyIdentifier extension of form [0] KeyIdentifier where the value of the KeyIdentifier field is taken from the value of the SubjectKeyIdentifier extension of the certificate issuer. The extension MUST be marked non-critical.

8.11.7 Additional ASN1 Definitions

The [CSEP] object identifier arc has been allocated by the IANA as follows:

```
csep OBJECT IDENTIFIER ::= { iso(1) identified-organizations(3) dod(6) internet(1) private(4)
  enterprise(1) 40732 }
```

8.11.7.1 SEP 2 Device Type Assignments

The deviceType is included in the CertificatePolicy extension of the Device Certificate and its issuing chain of CA certificates. It may be used for authorization purposes as described in Section 8.2.2.3.3.

```
deviceType OBJECT IDENTIFIER ::= { csep 1 }

id-SEP 2-dev-genericSEP 2Device OBJECT IDENTIFIER ::= { deviceType 1 }
-- used for most devices

id-SEP 2-dev-mobile OBJECT IDENTIFIER ::= { deviceType 2 }
-- used in addition to genericSEP 2Device to identify "mobile" SEP 2
-- entities (may be homed to multiple ESI domains)

id-SEP 2-dev-postManufactureSEP 2 OBJECT IDENTIFIER ::= { deviceType 3 }
-- used in device certs issued post-manufacture
```

8.11.7.2 SEP 2 Policy Assignments

One or more of SEP 2Policy OIDS MAY be included in the Device Certificate and its issuing chain of CA certificates.

```
SEP 2 Policy OBJECT IDENTIFIER ::= { csep 2 }

id-SEP 2-po-device-auth-test OBJECT IDENTIFIER ::= { SEP 2 Policy 1 }
-- MUST be included in test certificates

id-SEP 2-po-selfsigned-client OBJECT IDENTIFIER ::= { SEP 2 Policy 2 }
-- MUST be included in SEP 2 self-signed certificates
```

2009 id-SEP 2-po-service-provider OBJECT IDENTIFIER ::= { SEP 2 Policy 3 }
 2010 -- MUST be included in commercial certificates issued to
 2011 -- service providers for SEP 2 purposes.

2012 Id-SEP 2-po-bulk-cert OBJECT IDENTIFIER ::= { SEP 2 Policy 4 }
 2013 -- MUST be included in bulk-issued certificates (e.g.,
 2014 -- where the private key is generated off the device by the
 2015 -- issuing CA

2016 8.11.7.3 **HardwareModuleName**

2017 Excerpted from [RFC 4108]:

2018 id-on-hardwareModuleName OBJECT IDENTIFIER ::= {
 2019 iso (1) identified-organizations(3) dod(6)
 2020 internet(1) security(5) mechanisms(5) pkix(7) on(8) 4 }

2021 HardwareModuleName ::= SEQUENCE {
 2022 hwType OBJECT IDENTIFIER,
 2023 hwSerialNum OCTET STRING }

2024 The hwType field is assigned from the manufacturer's own OID arc according to its own policies. The
 2025 OID MUST be unique for each different manufacturer's device model and / or type. The manufacturer's
 2026 device type is NOT the same as the SEP 2 device type OID; instead it represents a single specific
 2027 product from a specific manufacturer.

2028 The hwSerialNum field is an unstructured field that the manufacturer should assign according to its own
 2029 policies and SHOULD be related to the serial number or other identifier on the device's external
 2030 physical label.

2031 The combination of hwType and hwSerialNum MUST be unique.

2032 Example:

2033 vendor1Devices OBJECT IDENTIFIER ::= { vendor1 13 }
 2034 meterNicV1 OBJECT IDENTIFIER ::= { vendor1Devices 1 1 }
 2035
 2036 HardwareModuleName = {
 2037 OID:1.3.6.1.4.1.99999.13.1.1, -- Vendor1 MeterNic V1
 2038 OCTET STRING: 0xa43218800 } -- Serial Number 44075-943936

2039 8.11.8 **Certificate Profiles**

2040 The certificates listed here have the normal [RFC 5280] format and the descriptions for all the fields
 2041 listed in the subsequent sections are described in [RFC 5280]. The absence of a field in the certificate
 2042 description is simply for conciseness and does not imply its absence in the certificate. In particular, the
 2043 Issuer Name is omitted in most of the certificate descriptions as [RFC 5280] requires it to be identical to
 2044 the Subject Name of the issuing certificate.

2045 By specification, [RFC 5280] certificates are encoded using the ASN1 Distinguished Encoding Rules.
 2046 For management or transmission purposes, they MAY be sent as ASN1 Distinguished Encoding Rules
 2047 (a file or stream of octets) or BASE64 encoded and possibly armored (i.e., "Privacy-Enhanced Mail
 2048 (PEM) format").

2049 SEP 2 devices MUST accept unexpected (not listed in this profile) certificate extensions and MUST
 2050 silently ignore non-critical unrecognized certificate extensions. Per [RFC 5280], devices MUST reject
 2051 any certificate containing unrecognized critical certificate extensions.

2052 8.11.8.1 **Root Certificate**

2053 A SERCA instance (e.g., contracted for CA operation) SHALL have exactly one self-signed certificate.
 2054 There MAY be more than one SERCA instance.

2055 The SERCA certificates are the root of trust for the Manufacturing PKI. They MUST contain the
 2056 extensions described below and MUST have the name form as described. They SHOULD NOT contain
 2057 any additional extensions.

- 2058 • Issued by: Self-signed
- 2059 • Issuer Name: O=Smart Energy, CN=SEP 2 Root, serialNumber=<n>
- 2060 • Subject Name: O=Smart Energy, CN=SEP 2 Root, serialNumber=<n>
- 2061 • Extensions
 - 2062 ○ certificatePolicy: critical; 1:anyPolicy
 - 2063 ○ keyUsage: critical; keyCertSign, crlSign
 - 2064 ○ basicConstraints: critical; cA=true, pathLen absent (unlimited)
 - 2065 ○ subjectKeyIdentifier: Section 8.11.6

2066 Note: The root certificate is primarily a container for a Trust Anchor.

2067 8.11.8.2 **Manufacturing Hierarchy Certificates**

2068 8.11.8.2.1 **MCA Certificate**

2069 MCA certificates MUST contain the extensions described below and MUST have at least the O and CN
 2070 components of the Subject Name as described. They SHOULD comply with the name form as described
 2071 below. They SHOULD NOT contain any additional extensions.

- 2072 • Issued by: SERCA
- 2073 • Subject Name: C=<country>, O=<Manufacturing Org>, CN=SEP 2 MCA,
 2074 serialNumber=<num>
- 2075 • Extensions:
 - 2076 ○ certificatePolicy: critical; at least one SEP 2 device type Identifier OID as a
 2077 policyIdentifier
 - 2078 ○ keyUsage: critical; keyCertSign
 - 2079 ○ basicConstraints: critical; cA=true, pathLen=1
 - 2080 ○ subjectKeyIdentifier: Section 8.11.6
 - 2081 ○ authorityKeyIdentifier: Section 8.11.6

2082 8.11.8.2.2 **MICA Certificate**

2083 MICA certificates MUST contain the extensions described below and MUST have at least the O and CN
 2084 component of the Subject Name as described. They SHOULD comply with the name form as described
 2085 below. They SHOULD NOT contain any additional extensions.

- 2086 • Issued by: SERCA or MCA
- 2087 • Subject Name: C=<country>, O=<Manufacturing Org>, CN=SEP 2 MICA,
 2088 serialNumber=<num>
- 2089 • Extensions:
 - 2090 ○ certificatePolicy: critical; at least one SEP 2 device type Identifier OID as a
 2091 policyIdentifier
 - 2092 ○ keyUsage: critical; keyCertSign
 - 2093 ○ basicConstraints: critical; cA=true, pathLen=0
 - 2094 ○ subjectKeyIdentifier: Section 8.11.6
 - 2095 ○ authorityKeyIdentifier: Section 8.11.6

2096 8.11.8.2.3 **Device Certificate**

2097 Device Certificates MUST contain the extensions described below. The Subject Name field MUST be
2098 empty. They SHOULD NOT contain any additional extensions. Except as modified by Section 8.11.4,
2099 the certificate is compliant with [IEEE 802.1AR].

2100 The device type identifier OID(s) MUST be selected from those present in the issuing certificate's
2101 certificatePolicy extension.

- 2102 • Issued by: SERCA or MICA
- 2103 • Subject Name: [EMPTY]
- 2104 • Extensions:
 - 2105 ○ certificatePolicy: critical; generally exactly one SEP 2 device type identifier OID as a
2106 policyIdentifier.
 - 2107 ○ subjectAlternativeName: critical; one GeneralName of type OtherName of
2108 hardwareModuleName (see Section 8.11.7.3 above for specific field values).
 - 2109 ○ keyUsage: critical; one or more of keyAgreement, digitalSignature.
 - 2110 ○ authorityKeyIdentifier: See Section 8.11.6.

2111 8.11.8.2.4 **Device Test Certificate**

2112 Device Test Certificates MUST contain the extensions described below. The Subject Name field MUST
2113 be empty. They SHOULD NOT contain any additional extensions. Except as modified by Section
2114 8.11.4, the certificate is compliant with [IEEE 802.1AR].

2115 The device type identifier OID(s) MUST be selected from those present in the issuing certificate's
2116 certificatePolicy extension.

2117 Note: This is the same format as the Device Certificate with the exception of an additional id-SEP 2-po-
2118 device-auth-test as policyIdentifier in the certificatePolicy extension.

2119 8.11.8.3 **Other Certificates**

2120 There are a few other certificates that a SEP 2 device may see.

2121 8.11.8.3.1 **Additional Certificates for SEP 2 Native Devices**

2122 8.11.8.3.1.1 *General Considerations*

2123 In addition to the certificates described in Section 8.11.8.2.3 above, native devices MAY include
2124 certificates (and their related key pairs) issued outside of the Manufacturing PKI. The provision of such
2125 certificates and support for them is OPTIONAL.

2126 A manufacturer MAY contract with any reputable Certificate Authority for the issuance of non-SEP 2
2127 Manufacturing certificates incorporating RSA public keys. The purpose for doing so is to provide
2128 backwards-compatible support to client devices, software and systems that may not yet support Elliptic
2129 Curve.

2130 Non-SEP 2 certificates MAY be placed in the device at one of two times:

- 2131 • During manufacture, either as a complete credential incorporating both private key and
2132 certificate, or as a certificate issued for a public key generated by the device.
- 2133 • During customer install as an over-the-net installation assuming connection to the Internet.

2134 For the latter approach, the specific protocol used to install the certificate is vendor-dependent and
2135 should assume only normal Internet connectivity. Specifically, it should not depend on any "proxy" or
2136 third-party assistance within the customer's home or the service provider's back-office.

2137 8.11.8.3.1.2 *Certificate Structure and Certificate Chain Considerations*

2138 The certificate installation for these additional certificates MUST include the complete chain of
 2139 certificates needed to validate the certificate, including the root of trust certificate for the chain.

2140 The certificates MUST contain an RSA 2048-bit public key, and MUST be signed using SHA256 with
 2141 RSA. They SHOULD contain an expiration date not later than December 31st, 2028. As the provision of
 2142 RSA certificates is considered a transition mechanism, this is an appropriate way to phase out the use of
 2143 such certificates and the RSA signature algorithms. Intermediate certificates in the chain SHOULD
 2144 contain an expiration date not earlier than December 31st, 2020.

2145 The SEP 2 device SHOULD permanently stop using the certificate and related private key upon
 2146 certificate or chain expiration if the device has a mechanism for determining time and date.

2147 The Subject Name form for the Device Certificate MAY be any form approved by the certificate issuer.
 2148 This specification recommends that the Subject Name be crafted as: "C=<Country of
 2149 certification>,O=<Manufacturer>, OU=<Device type>, UID=<Serial Number>". An RSA certificate
 2150 MUST NOT use a SubjectAlternativeName extension in place of a SubjectName unless it is identical to
 2151 the SubjectAlternativeName extension in the device's Device Certificate.

2152 8.11.8.3.1.3 *Use Case*

2153 The above-described certificates are targeted for use when an external client connects to a server using
 2154 TLS. The TLS negotiation and response is thus:

- 2155 • When a SEP 2 device containing an additional RSA certificate receives a TLS Client Hello
 containing only a supported RSA cipher suite, it SHOULD answer any certificate negotiation
 with the non-SEP 2 RSA certificate.
- 2158 • If a SEP 2 device receives a TLS Client Hello containing both EC and RSA supported cipher
 suites, it SHOULD respond using its EC Device Certificate, regardless of cipher suite
 preference order.
- 2161 • If a SEP 2 device without an additional certificate receives a TLS Client Hello containing only
 an RSA cipher suites, it MUST reject the connection with the appropriate code.

2163 8.11.8.3.2 **Self-Signed Client Certificate**

2164 Any application designed to talk to native SEP 2 products or applications implementing SEP 2 functions
 2165 may create a credential for itself consisting of a self-signed [RFC 5280] certificate. The application
 2166 generates the key pair and binds the public key in a certificate. The credential privilege is installed in the
 2167 SEP 2 device by either passing the certificate or a fingerprint of the certificate along with the credential
 2168 privileges to the appropriate SEP 2 application using an enrolment protocol (not specified here) or other
 2169 mechanism (e.g., web portal mechanism for moving from unsecure to secure).

2170 A SEP 2 device or application acting in a client role SHALL reject a self-signed certificate if presented
 2171 by the server. A SEP 2 device or application acting in a server role MAY legitimately receive a Self-
 2172 signed Certificate from a potential client. The server SHALL verify that the Self-signed Certificate
 2173 follows the format described below and SHALL reject the certificate if there are any discrepancies. A
 2174 server MUST NOT treat a Self-signed Certificate received through a TLS Handshake as corresponding
 2175 to a root CA, unless the public key carried in the Self-signed Certificate is the same as one of the pre-
 2176 provisioned roots.

- 2177 • Issued by: Self-signed
- 2178 • Subject Name: Any – suggested: O=<application name>,CN=<12digit random hex string>
- 2179 • Issuer Name: Identical to Subject Name
- 2180 • Validity: notBefore: time of issue; notAfter: maximum of time of issue plus 3 years.

- 2181 • Subject Public Key and Signature: SHOULD be EC P-256 and SHA256withECDSA, MAY be
2182 RSA 2048 and SHA256withRSA.
2183 • Extensions
2184 ○ keyUsage: critical; at least digitalSignature, others as appropriate.
2185 ○ certificatePolicy: critical; at least one policyIdentifier:id-SEP 2-po-selfsigned-client.

2186 8.11.8.3.3 **Generic Client Certificates for non-SEP 2 Entities**

2187 In addition to application issued self-signed certificates, an application may use a certificate issued by a
2188 global or local CA as an application credential for use with the SEP 2 protocol. As with a self-signed
2189 certificate, the credential privilege is installed in the SEP 2 device or application by either passing the
2190 certificate or a fingerprint of the certificate along with the credential privileges. At a later date, SEP 2
2191 may specify further uses for certificates not issued under the SEP 2 certificate hierarchies.

2192 There are no differences between the uses for a self-signed client certificate and "other" client
2193 certificates. The major difference is simply how the certificates are issued or who issues them. One
2194 example of an "other" client certificate would be an email or SSL client certificate issued by one of the
2195 well-known Certificate Authorities.

2196 Client certificates by the SEP 2 definition are those that do not contain a basicConstraints extension.

2197 8.11.8.3.4 **Generic Server Certificates for non-SEP 2 Entities**

2198 SEP 2 devices or applications may need to connect to TLS servers secured by server certificates issued
2199 by global or local CAs not affiliated with [CSEP]. For interoperability with this profile, those certificates
2200 MUST meet the following requirements:

- 2201 • MUST contain either an RSA 2048-bit public key or a EC P-256 public key.
2202 • MUST be signed either using either the `SHA256withRSA` or `SHA256withECDSA` algorithms.

2203 In addition, the SEP 2 client device MUST have a valid root of trust for the server certificate. This can
2204 be installed at manufacture, or installed by the operator or owner after the device is operational. The
2205 methods for the installation of these additional roots of trust are vendor specific.

2206 The connectivity of SEP 2 devices to external devices is subject to further specification.

2207 8.11.9 **Device Requirements**

2208 8.11.9.1 **Private-key Protection**

2209 Device manufacturers SHOULD ensure that, once installed, private keys cannot be exported from the
2210 device.

2211 8.11.9.2 **Trusted Key Store Protection**

2212 Device manufacturers SHALL ensure that the current active Smart Energy Root CA (SERCA)
2213 certificates or root public keys are embedded in the device at the time of manufacture and firmware
2214 upgrade. They SHALL ensure that, once installed, the Root CA public keys cannot be overwritten via an
2215 over-the-air action, except as a by-product of a successful firmware upgrade.

2216 8.11.9.3 **Certificate Usage**

2217 Device manufacturers SHALL ensure that a complete and valid Manufacturing PKI certificate chain
2218 (e.g., SERCA, MCA if any, issuing MICA if any, and Device Certificate) is embedded in the device at
2219 the time of manufacture.

2220 In an authentication exchange, the device SHALL supply a complete and valid chain comprising its own
2221 certificate and any intermediate CA certificate between the device and the root (i.e.; all certificates in its
2222 chain except its SERCA).

2223 8.11.9.4 **Trusted Certificate Store**

2224 A device meant to act as a server to non-SEP 2 entities MUST incorporate at manufacture a list of
2225 certificates for non-SEP 2 Root CAs recommended by [CSEP] for use as TLS trust anchors. These non-
2226 SEP 2 Root CAs typically have no association with [CSEP] or a SERCA.

2227 The [CSEP] shall publish such a list of certificates on its website and shall update the list no less often
2228 than annually. Manufacturers are responsible for ensuring the update of their manufacturing processes
2229 within 60 days of the publication of a new list. There shall be an initial list size of 10 non-SEP 2 Root
2230 CA certificates, increasing by one a year to a maximum of 20 non-SEP 2 Root CA certificates.

2231 Vendors SHOULD incorporate an update of the trust store as part of any firmware update where
2232 appropriate. Vendors MAY provide a mechanism for a consumer / customer to set the trust status of any
2233 Root CA certificate and to add and delete such certificates from the trust store of their owned device.

2234 8.11.10 **Certificate Verification**

2235 SEP 2 devices and applications MUST follow the procedure defined in [RFC 5280], Section 6 to verify
2236 certificates.

2237 Certificate verifiers MUST reject certificates that contain one or more unsupported critical extensions.

2238 Policy-mapping is not supported, and certificates containing policy mappings MUST be rejected.

2239 Policy-qualifiers are not supported. Therefore, issuers SHOULD NOT include policy qualifiers in
2240 certificates. However, verifiers SHOULD NOT reject certificates containing policy qualifiers unless
2241 there are other reasons to do so.

2242 Name-constraints are not supported and certificates containing name-constraints MUST be rejected.

2243 Any extension not listed by name within this document SHOULD NOT be included within a compliant
2244 certificate and, if included, MUST NOT be marked critical.

2245 8.11.10.1 **Additional Considerations for Serving SEP 2 to Non-native Entities**

2246 A SEP 2 device MAY implement OCSP for the sole purpose of validating non-SEP 2 certificates (e.g.,
2247 as received from clients and applications using generic or self-signed client certificates, or when
2248 contacting generic servers). In addition or in lieu of this, a SEP 2 device or application may use a white
2249 list or other access control list to determine acceptance of an offered client certificate from an external
2250 source. A SEP 2 device or application MUST NOT use OCSP for the purpose of attempting to validate
2251 Manufacturing PKI-issued certificates.

2252 As none of the Manufacturing PKI CAs issue either CRLs or run OCSP servers, no non-SEP 2 device or
2253 application may use CRLs or OCSP for the purpose of attempting to validate SEP 2 certificates.

2254 8.11.11 **Certificate Related Labeling Requirements**

2255 For the purposes of Access Control Lists and other human-readable actions, the device will be identified
2256 by the data listed in Section 8.3. These data may be used as appropriate to label a device or its
2257 packaging.

2258

9 Discovery

2259 Smart Energy Profile 2.0 specifies DNS-based methods for service discovery, resource discovery, and
2260 hostname to IP address resolution. A *service* is defined as an application instance uniquely identified by
2261 {host, port, protocol}, where protocol in this case is SEP 2 plus its underlying transport bindings (e.g.,
2262 HTTP(S)/TCP/IP). DNS-based Service Discovery (DNS-SD) [RFC 6763] is a conventional use of
2263 existing DNS name syntax and message and record formats (PTR, SRV, TXT) to discover instances of a
2264 given service within a given domain. In SEP 2.0, DNS-SD [RFC 6763] is used to describe the location
2265 of function sets and groups of resources by supplying the host, port and protocol of the supporting
2266 servers along with additional details provided by those servers.

2267 DNS-SD specifies that a DNS SRV and TXT record pair are used to describe a service instance. Both
2268 records have an identical Service Instance Name of the form "<Instance>.<Service>.<Domain>". The
2269 SRV record contains the hostname and port of the service, while the TXT record may contain additional
2270 variables (such as a relative path) in text form. A service plus a path forms a URI and can be used to
2271 locate a resource. A client discovers instances of a given service or resource type by sending a query for
2272 a DNS PTR record with the name "<Service>.<Domain>", which returns a set of zero or more Service
2273 Instance Names of DNS SRV/TXT record pairs for the requested service or resource type.

2274 Multicast DNS (mDNS) [RFC 6762] provides the ability to perform DNS-like queries on the local link
2275 in the absence of any conventional unicast DNS server. In addition, mDNS reserves the top-level DNS
2276 domain ".local." to name services that have link-local scope. mDNS employs link-local multicast
2277 addressing for requests and either multicast or unicast addressing for responses in support of service
2278 discovery. IPv6 address scoping is used to specify reachability and includes address types for global,
2279 site-local, and link-local addressing [RFC 4291].

2280 Extended Multicast DNS (xmDNS) [I-D XMDNS] extends the scope of mDNS beyond the local link
2281 through the use of site-local multicast requests and responses. In addition, xmDNS reserves the top-level
2282 DNS domain ".site." to name services that have site-local scope. The site-local multicast address
2283 FF05::FB is designated for Extended Multicast DNS and SHALL be used as the destination address for
2284 all xmDNS multicast requests and multicast responses. The reachability of this address is
2285 administratively defined and MAY span multiple sub-networks. SEP 2 SHALL use global addresses or
2286 Unique Local Addresses [RFC 4193] in the source address of xmDNS requests and responses. xmDNS
2287 [I-D XMDNS] is normative for SEP 2.

2288 Guidelines on the use of unicast responses SHALL be employed as described in the xmDNS draft unless
2289 noted otherwise. xmDNS requests from HAN devices that are mains powered SHALL use site-local
2290 multicast addressing to ensure that all sub-networks within the HAN are reachable and MAY request
2291 unicast responses. Battery operated devices in the HAN SHOULD use site-local multicast addressing for
2292 xmDNS requests and SHOULD request unicast responses. HAN devices making xmDNS requests that
2293 generate responses specific to the requesting HAN device (e.g., requests such as those requesting
2294 EndDevice servers where the particular HAN device is registered) SHALL employ site-local multicast
2295 destination addresses and SHALL request unicast responses. When a server receives a request with the
2296 QU bit set it SHALL return a unicast response.

2297 9.1 Service Instance

2298 A server SHALL assign a unique <Instance> label of up to 63-bytes in UTF-8 format for each DNS
2299 SRV / TXT record pair that it advertises. In order to avoid name conflicts, <Instance> names SHOULD
2300 begin with a meaningful substring followed by a hyphen '-' and end with the device's SFDI or other
2301 collision-resistant substring such as the low-order bits of an EUI-64 in text form (e.g., device-
2302 000001111114). Should a name conflict occur, a device SHALL assign itself a new name until conflicts

2303 are resolved. A conflict SHOULD be resolved by appending a decimal integer in parentheses to the
 2304 <Instance> (for example, `Name (2)` for the first conflict, `Name (3)` for the second conflict, etc.).
 2305 If a DNS SRV/TXT record pair is created to advertise a function set, its <Instance> SHOULD consist of
 2306 the corresponding string from the Subtype column of Table 9-2 followed by a hyphen and a collision-
 2307 resistant substring as defined above (e.g., `upt-00000111114`). When an SFDI is used as part of a DNS-
 2308 SD label, it SHALL be represented as 12 decimal digits including leading zeros (if any) as well as the
 2309 checksum digit and SHALL NOT include embedded hyphens.

2310 9.2 Service Name

2311 The Service Name used with SEP 2 DNS-SD SHALL be `smartenergy` and has been registered
 2312 appropriately with IANA [IANA SN].

2313 The <Service> portion of a Service Instance Name consists of the Service Name preceded by an
 2314 underscore `_` and followed by a period plus a second DNS label specified by SEP 2 as `_tcp`.

2315 Thus, a valid Service Instance Name example would be:

2316 `device-00000111114._smartenergy._tcp.site.`

2317 Where `device-00000111114` is the <Instance> portion (described above), `smartenergy` is the Service
 2318 Name, `tcp` is the transport protocol, and `"site."` is the <Domain> portion (xmDNS is being utilized in
 2319 this example).

2320 9.3 TXT Record

2321 This sub-section specifies the format of the TXT record to be used in conjunction with SEP 2 DNS-SD.
 2322 The Smart Energy Profile 2.0 application SHALL use a single TXT record format. The following table
 2323 describes the supported TXT record parameters for Smart Energy Profile 2.0.

2324 **Table 9-1 TXT record parameters.**

Key=Value	Example
<code>txtvers={#}</code>	<code>txtvers=1</code>
<code>dcap={relative reference to DeviceCapabilities}</code>	<code>dcap=/dcap</code>
<code>path={relative reference to the function set} corresponding to the specified subtype name</code>	<code>path=/upt</code>
<code>https={port }</code>	<code>https=443</code>
<code>level={schema extensibility level indicator}</code>	<code>level=S0</code>

2325 `txtvers` SHALL be the first key in the TXT record. For version 2.0 of the Smart Energy Profile, the
 2326 value of the `txtvers` key SHALL be 1. If it is found in a response to be other than 1, the TXT record
 2327 SHALL be ignored.

2328 The `txtvers` key SHALL be present with a non-empty value. Clients SHALL silently discard TXT
 2329 records with `txtvers` keys that are not present with a non-empty value of 1.

2330 Unknown `key=value` pairs in a response SHALL be ignored.

2331 The `dcap` key SHALL provide the relative reference used to locate the device's DeviceCapability
 2332 resource and SHALL include the leading slash of the given path.

2333 The `dcap` key SHALL be present with a non-empty value. Clients SHALL silently discard TXT records
 2334 with `dcap` keys that are not present with a non-empty value representing the URI for the server's
 2335 DeviceCapability resource.

2336 The `path` key is used in responses to subtype queries (see below) and SHALL provide the relative
 2337 reference used to locate the base path of a specified function set or resource, including the leading slash
 2338 '/' of the given path.

2339 The `path` key SHALL be omitted in response to service name queries. The `path` key SHALL be present
 2340 with a non-empty value representing the URI satisfying the subtype query. Clients SHALL ignore `path`
 2341 keys included with service name query responses where the `path` key is supplied with no value or
 2342 present with an empty value.

2343 The `https` key is used to indicate whether or not the function set requires a secure (HTTP over TLS)
 2344 connection. If a value is present for this key in the TXT record, a client SHOULD locate the
 2345 corresponding function set or resource using a secure connection to the specified port in order to avoid a
 2346 re-direct.

2347 The `https` key MAY be present with no value or present with an empty value if HTTPS is supported for
 2348 the query using the default port of 443. Servers supporting HTTPS using a non-default port SHALL
 2349 indicate the port number by including the `https` key with a non-empty value representing the supported
 2350 HTTPS port number. Clients SHALL use HTTP for the service if the `https` key is not present. Clients
 2351 SHALL use HTTPS using the default port for the service if the `https` key is present with no value or
 2352 present with an empty value. Clients SHALL use HTTPS using the port number indicated if the `https`
 2353 key is present with a non-empty value.

2354 The `level` key is used to indicate the Extensibility Level of the schema for the specified function set (see
 2355 Section 7.7). The default value SHALL be the supported Extensibility Level of the server. If “-S[i]” is
 2356 provided in the `level` key, the server does not support extensions to the schema. If “+S[i]” is provided
 2357 in the `level` key, the server SHALL support either of “-S[i]” or “+S[i]” as negotiated through the HTTP
 2358 Accept header.

2359 The `level` key SHALL be present with a non-empty value. Clients SHALL silently discard TXT records
 2360 with `level` keys that are not present with a non-empty value representing the servers schema level
 2361 extensibility indicator.

2362 9.4 Subtype Queries

2363 Subtype names act as filters that return the SRV / TXT record pairs describing a given function set. For
 2364 example, if a device such as an electricity meter also serves gas metering data via mirroring, that device
 2365 will register two subtype names; one for providing metering data and one for the capability to receive
 2366 metering data to mirror. A client device can search for instances of a given function set by first
 2367 performing a subtype query and then interrogating the Device Capabilities URIs (contained in the
 2368 returned TXT records) to determine the URIs for that function set. Alternately, it may use a `path`
 2369 returned directly by the server as described below.

2370 SEP 2 uses an extended form of subtype query in order to support fine-grained resource discovery and
 2371 conserve bandwidth. Using this method, a server will return a separate SRV / TXT record pair for each
 2372 function set that it supports. The name of the DNS-SD PTR record is equivalent to the query argument
 2373 and the value of the PTR record is the Service Instance Name of an SRV / TXT record pair matching the
 2374 query.

2375 A server SHALL register a PTR record with a subtype name as defined below for each function set that
 2376 it advertises for discovery. In addition, the server SHALL register a unique SRV / TXT record pair with

2377 an <Instance> as defined in Section 9.1 for each function set that is referenced by the subtype PTR
 2378 record.

2379 All SRV records registered on a given device SHALL contain identical values. The port value contained
 2380 in the SRV record SHALL be specified for the default (http) scheme. If a secure connection is required
 2381 for the function set or resource, then the `https` key SHALL be present in the TXT record as specified in
 2382 Section 9.3.

2383 The server SHALL register exactly one PTR record with the name "`_smartenergy._tcp.site.`". The SRV
 2384 / TXT record pair referenced by this PTR record refers to the server itself. Enumerating the set of all
 2385 SEP 2 servers in the HAN is STRONGLY DISCOURAGED except for diagnostic purposes. Instead,
 2386 devices SHOULD use subtype queries as described in this section to enumerate SEP 2 function sets.

2387 The TXT record of the SRV / TXT record pair referenced by a subtype PTR record SHALL conform to
 2388 the definition given in Section 9.3 and SHALL contain the relative reference for the base path of the
 2389 function set that corresponds to the specified subtype. The `key=value` pairs other than `path` and `https`
 2390 SHOULD be identical for all TXT records registered on a single device.

2391 Subtype names are comprised of a subtype string, followed by "`_sub._smartenergy._tcp.site.`".
 2392 Subtype strings SHALL NOT begin with an underscore (see Table 9-2). For example, the subtype name
 2393 for the meter usage point function set shall be composed as "`upt._sub._smartenergy._tcp.site.`". Other
 2394 subtype names SHALL (except as noted below) be composed per the meter usage point example.

2395 The following table lists the defined service subtype strings and their corresponding SEP 2 function sets.

2396 **Table 9-2 Service subtype strings and their corresponding SEP 2 function sets.**

Subtype	Device Capabilities Field Name	SEP 2 Function Set
bill	CustomerAccountLink	Billing
derp	DERProgramLink	Distributed Energy Resources
dr	DemandResponseProgramLink	Demand Response / Load Control
edev	EndDeviceLink	End Device
file	FileLink	File Download
msg	MessagingProgramLink	Messaging
mup	MirrorUsagePointLink	Metering Mirroring
ppy	PrepaymentLink	Prepayment
rsp	ResponseSetLink	Response
sdev	SelfDeviceLink	Self Device
tm	TimeLink	Time
tp	TariffProfileLink	Pricing
upt	UsagePointLink	Metering

2397 To promote search efficiency, servers that support the End Device function set SHALL register a unique
 2398 PTR, SRV, and TXT record for each remote device that they support. In this case, the subtype name
 2399 SHALL consist of the string `edev`, concatenated with a hyphen and the remote device's SFDI, (see
 2400 Section 8.3.2) and followed by "`_sub._smartenergy._tcp.site.`". For example, a server having an End
 2401 Device resource for the remote device with SFDI 222222222228 would register a subtype PTR record
 2402 named "`edev-222222222228._sub._smartenergy._tcp.site.`".

2403 The <Instance> portion of the Service Instance Name of the associated SRV / TXT record pair for this
2404 subtype PTR SHOULD consist of three parts, concatenated by hyphens. The first part SHALL consist
2405 of an identifier unique across End Device instances on this server. The second part SHALL be the `eudev`
2406 subtype string. The third part SHOULD be the server's SFDI. For example, a server having the SFDI
2407 "00000111114" would register SRV and TXT records having the Service Instance Name "`127-eudev-`
2408 `00000111114._smartenergy._tcp.site.`", where `127` is an arbitrary decimal number unique to this
2409 specific End Device instance on this server. This name would be referenced by the subtype PTR record
2410 described in the example above. The TXT record with this name would contain the path to the End
2411 Device resource of the remote device.

2412 9.5 Discovery Procedure

2413 This section provides a walkthrough of the process a client would use to find a resource of interest using
2414 the appropriate portions of this specification.

2415 Link-layer joining is not covered in this specification. Refer to the appropriate specification for details
2416 regarding joining the appropriate data link-layer network.

2417 The following sequence demonstrates the discovery process using DeviceCapabilties (as opposed to
2418 Function Set Assignments). This sequence assumes some knowledge either from other specifications or
2419 from other sections in this document, therefore these may need to be read before this sequence is fully
2420 understood.

- 2421 1) Use xmDNS/DNS-SD to locate the servers with the function sets of interest
2422 (See [I-D XMDNS], [RFC 6763]).
- 2423 2) For each server do the following:
 - 2424 a) Establish TLS session if required (see Section 8).
 - 2425 b) GET the Device Capabilities resource (See Section 10.2).
 - 2426 c) Look for the desired function set in the Device Capabilities resource.
 - 2427 d) If there is an entry in Device Capabilities it will contain a URI of the entry point for that
2428 function set.
 - 2429 e) Alternately, use the `path` returned in the subtype query response (See Section 9.4).
- 2430 3) Determine which of the discovered resources are of interest. This depends on the resource and
2431 other outside factors (e.g., if the resource is "metering", then the meter of interest might be the
2432 one that has the Premises Aggregation Point attribute set to true).

2433 It should be noted that clients SHALL dynamically discover the URI(s) of the resource(s) of interest, as
2434 the URI(s) MAY vary from server to server and MAY occasionally vary over the lifetime of a given
2435 server. Clients SHALL rediscover URIs upon notification of the server DNS-SD record change or a
2436 request fails with a 404 (Not Found) error. Clients SHALL in the case of redirects, follow the guidance
2437 in the HTTP specification [RFC 2616].

2438

2439 10 Support Resources

2440 This section defines resources and function sets that provide operational information to the end devices
 2441 of an SEP 2 network or provide those end devices with services to manage and support their operation.
 2442 The section begins with a general description of how each of sub-sections of this and the following two
 2443 sections are organized.

2444 10.1 Resource Section Outlines

2445 This section gives the reader a basic understanding of the outline of each of the sections describing the
 2446 resources of SEP 2. The intent is that each of these resources can be implemented on independent
 2447 servers or grouped to coexist on a single server. Keep in mind that these resources or Function Sets are
 2448 defined in three documents, the SEP 2 Application Specification (this document), the SEP 2 XML
 2449 Schema and the SEP 2 WADL described in document [ZB 13-0201]. All three need to be consulted to
 2450 get a full understanding of SEP 2.

2451 The Resource sections follow a standard "template", sometimes modified based on unique
 2452 circumstances. Each section is outlined as follows:

- 2453 1) Overview
- 2454 2) List Ordering
- 2455 3) Application Guidelines / Behavior
- 2456 4) LogEvents

2457 Each of these sections is discussed in more detail below.

2458 10.1.1 Overview

2459 Contains a brief, three-to-four sentence, informative description of the functionality provided by the
 2460 function set or resource. It does not state normative requirements.

2461 10.1.2 List Ordering

2462 This section provides guidance regarding: how resource elements and attributes are used to support
 2463 unique ordering of resources within lists.

2464 **Table 10-1 Function Set List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
Resource	Resource.elementA (descending or ascending)	Resource.elementC (descending or ascending)	NA
Sub-Resource	Sub-Resource.elementD (descending or ascending)	Sub-Resource.elementB (descending or ascending)	NA

2465 10.1.3 Application Guidelines / Behavior

2466 This section describes the normative, high level behavior of the function set and defines how the
 2467 function set resources are used by clients and servers to accomplish the goals of the function set.
 2468 Normative definitions of resource elements and attributes can be found in the XML schema described in
 2469 [ZB 13-0201]. This section contains non-trivial information about resources that is too complex to be
 2470 represented in the XML schema and non-functional requirements for clients and servers (e.g., minimum
 2471 / recommended support for resource instances).

2472 This sub-section may also include application guidelines germane to sleepy HAN devices.

2473 10.1.4 **LogEvents**

2474 This sub-section includes definitions of all LogEvents that may be raised by the function set. Note that
 2475 function sets locally define their own LogEvent cardinal values, which typically will not be unique
 2476 across function sets. The function set specific LogEvent codes are combined with a function set
 2477 identifier to produce a unique code.

2478 All LogEvent definitions are presented in the format shown by Table 10-2.

2479 **Table 10-2 Example LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
LE_EXAMPLE_0	0x00	Normative text defining when the event should be generated.
LE_EXAMPLE_1	0x01	Normative text defining when the event should be generated.

2480 10.2 **Device Capabilities Function Set**2481 10.2.1 **Overview**

2482 The DeviceCapability resource enumerates the function sets supported by a device and can be used by
 2483 clients to discover location information for the enumerated function sets.

2484 10.2.2 **List Ordering**

2485 This resource does not contain any lists so no page ordering is specified.

2486 10.2.3 **Application Guidelines / Behavior**

2487 Smart Energy 2 clients locate HAN services by performing DNS Service Discovery (DNS-SD) queries
 2488 to the HAN. A query may be issued non-specifically for any Smart Energy 2 device, or a Smart Energy
 2489 2 device supporting a specific function set. Successful DNS-SD queries return the URI to the matching
 2490 server's DeviceCapability resource. The client is then free to further access the function sets enumerated
 2491 within the DeviceCapability resource.

2492 Clients MAY query this resource to determine what resources are available on the given server. The
 2493 resources a server exposes MAY be determined by the access rights of the client on this server. Servers
 2494 MAY hide resources that a client does not have access rights to. For an alternative way to locate
 2495 resources see the sections on End Device and Function Set Assignments.

2496 A resource serving device (i.e., all servers) SHALL implement the DeviceCapability resource.

2497 10.2.4 **LogEvents**

2498 There are no LogEvents generated by this function set.

2499 10.3 **Self Device Resource**2500 10.3.1 **Overview**

2501 The SelfDevice resource provides an interface for servers to publish general information about
 2502 themselves (e.g., SFDI, LFDI, software version numbers).

2503 10.3.2 **List Ordering**

2504 This resource does not contain any lists so no ordering is specified.

2505 10.3.3 **Application Guidelines / Behavior**

2506 If a server hosts resources pointed to from a SelfDevice instance (e.g., DeviceInformation via
 2507 DeviceInformationLink, PowerStatus via PowerStatusLink), those resources SHOULD be restricted to

2508 read-only access. That is, even if the SEP 2 WADL allows PUT, POST, or DELETE access, those
 2509 HTTP methods SHOULD be blocked (or restricted to access by the server itself, i.e. through a loop-back
 2510 interface).

2511 Exceptions to the above recommendation are as follows:

- 2512 • Self Device servers MAY allow read / write access to the resource pointed to by
 2513 ConfigurationLink.

2514 10.3.4 LogEvents

2515 **Table 10-3 Self Device LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
SE_INVALID_WRITE_TO_SELF	0x00	MAY be generated in response to PUT, POST, DELETE attempts on the SelfDevice instance or any of its write-protected sub-resources.

2516 **10.4 End Device Resource**

2517 **10.4.1 Overview**

2518 The EndDevice function set provides interfaces to exchange information related to particular client
 2519 device(s). This may include both general information about a client (e.g., SFDI, software version
 2520 numbers) and information specific to the relationship between a client and the server where the
 2521 EndDevice resource is hosted (e.g., subscriptions, function set assignments, registration). The major
 2522 resources are listed below:

- 2523 • EndDeviceList
 2524 • EndDevice

2525 EndDeviceList provides the list of all EndDevice resources hosted by the given server. Each EndDevice
 2526 resource corresponds to a particular client device for which the server is maintaining persistent
 2527 information. On an EndDevice function set server, an EndDevice resource may be created as part of an
 2528 out-of-band registration process. EndDevice resources may also be created on a POST to the
 2529 EndDeviceList.

2530 **10.4.2 List Ordering**

2531 **Table 10-4 End Device List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
EndDevice	SFDI (ascending)	href (ascending)	N/A

2532 **10.4.3 Application Guidelines / Behavior**

2533 Servers MAY present the EndDeviceList differently based on the credentials of the requesting client.
 2534 For example, even if a server contains the EndDevice resources for five registered clients, any one
 2535 registered client that GETs the EndDeviceList may receive a list resource containing only the one
 2536 EndDevice resource corresponding to that client. Similarly, an unregistered client that GETs the
 2537 EndDeviceList may receive an empty EndDeviceList list resource. However, clients SHALL NOT rely
 2538 on this filtering behavior, as it is not mandatory server behavior; therefore, clients SHALL support the
 2539 normal paging mechanism for List-type resources.

2540 If a server hosts resources pointed to from an EndDevice instance (e.g., DeviceInformation,
 2541 DevicePowerStatus) that allow PUT, POST, or DELETE access, those HTTP methods SHOULD be
 2542 restricted to the client device represented by the given EndDevice.

2543 Clients MAY update their EndDevice and related status resources at regular intervals, or when the
 2544 relevant information changes, to the appropriate sub-resources of an EndDevice instance that
 2545 corresponds to the client.

2546 Clients SHALL NOT POST a new EndDevice instance to a server's EndDeviceList if that
 2547 EndDeviceList already contains an EndDevice instance for the client.

2548 It is RECOMMENDED that servers remove records after 72 hours of no activity from the corresponding
 2549 client. It is RECOMMENDED that clients POST at least once every 48 hours.

2550 Servers SHOULD delete or prevent duplicate EndDevice instances for the same client from appearing in
 2551 its EndDeviceList. For instance, certificate information transmitted as part of TLS may be used to
 2552 uniquely identify a particular client and prevent it from POSTing multiple EndDevice instances.

2553 Servers MAY remove EndDevice instances and / or their subordinate resources at any time, for any
 2554 reason. Clients that have cached URIs for their EndDevice instance and / or subordinate resources but
 2555 are no longer able to access those resources SHOULD attempt to rediscover the new locations of those
 2556 resources, recognizing that they MAY have been permanently deleted by the server (e.g., as part of an
 2557 out-of-band de-registration process).

2558 When a Server receives an EndDevice resource via a POST to its EndDeviceList, the Server SHOULD
 2559 NOT modify any of the Link attributes inherited from AbstractDevice if the Link contains a fully
 2560 qualified URI. If a POSTed Link does not contain a fully qualified URI, the Server MAY allocate local
 2561 storage for that subordinate resource and populate the EndDevice resource with a Link pointing to the
 2562 allocated resource, or the Server MAY choose not to support that particular subordinate resource, if
 2563 allowed by the SEP 2 schema, in which case that particular Link attribute would not appear in the newly
 2564 created EndDevice resource.

2565 Clients that POST EndDevice resources to the EndDeviceList of a Server SHOULD NOT populate Link
 2566 attributes in that EndDevice resource with fully qualified URIs that point to resources on the Server
 2567 itself. Servers would typically allocate storage and URIs themselves.

2568 10.4.4 LogEvents

2569 **Table 10-5 End Device LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
LE_UNAUTHORIZED_WRITE_TO_END_DEVICE	0x00	MAY be generated in response to PUT, POST, and DELETE attempts by an unauthorized device on an EndDevice instance or any of its protected sub-resources.

2570 **10.5 Function Set Assignments**

2571 **10.5.1 Overview**

2572 The FunctionSetAssignments resource defines collections of references to function set instances. These
 2573 collections are used by the End Device resource for indicating to devices the resources that are to be
 2574 used for specific purposes. For example, a service provider may wish that all the participants of a

2575 particular program use a given DRLC function set instance, Time resource, and Pricing function set
 2576 instance. A FunctionSetAssignments resource contains references to the function set instances that are to
 2577 be used by the assigned device(s).

2578 **10.5.2 List Ordering**

2579 **Table 10-6 Function Set Assignments List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
FunctionSetAssignments	mRID (descending)	NA	NA

2580 **10.5.3 Application Guidelines / Behavior**

2581 The Function Set Assignments function set is a means to indicate collections of particular instances of
 2582 function sets. Each instance of the Function Set Assignments function set contains lists of references to
 2583 particular function set instances. Service providers, HEMS, and possibly other entities MAY use this
 2584 service. They MAY create these instances based on a HAN asset class / category, service provider
 2585 program, or any other criteria.

2586

2587 Specific instances of FunctionSetAssignments resources are defined out of band and the methods for
 2588 getting the information into the instances is outside the scope of this specification. Server use of this
 2589 function set is OPTIONAL. Clients that act on events SHALL determine if they are assigned into
 2590 Function Set Assignments. Clients MAY be assigned multiple Function Set Assignments. Multiple
 2591 Clients MAY be assigned the same Function Set Assignment. If a server supports Function Set
 2592 Assignments it SHALL support a minimum of 1 Function Set Assignments for each HAN device
 2593 registered to the server. A server SHOULD support 3 Function Set Assignments for each HAN device.
 2594 Clients SHALL support at least 6 function set instances (e.g., two DemandResponseProgram instances,
 2595 one Time instance, one TariffProfile instance, and two MessagingProgram instances) assigned through 1
 2596 or more Function Set Assignments.

2597

2598 If a FunctionSetAssignments instance contains references to time-responsive function sets, it MUST
 2599 also include a reference to a Time resource.

2600

2601 Clients SHALL identify which Function Set Assignments apply to them by querying the
 2602 FunctionSetAssignments resource within their End Device function set instance (see the End Device
 2603 section for more details).

2604

2605 Clients SHALL periodically poll their group assignments under their EndDevice resource (e.g.,
 2606 /edev/{#}/fsa), and the corresponding Function Set Assignments resource (e.g. /fsa/{#}), or SHALL
 2607 subscribe to them to monitor for changes. Client devices that do not subscribe SHALL query at least
 2608 once every 24 hours but SHALL NOT query more than once per hour.

2609 **10.5.4 LogEvents**

2610 There are no LogEvents generated by this function set.

2611 **10.6 Subscription / Notification Mechanism**

2612 **10.6.1 Overview**

2613 This section describes the design and use of resources that support a generic, lightweight subscription /
 2614 notification mechanism for use throughout the specification. This mechanism may be useful when a
 2615 client wishes to quickly learn of changes to a resource on a server.

2616 The following text further clarifies the roles of the subscription and notification resources.

2617 10.6.2 **List Ordering**

2618 Notification resources typically are not exposed such that they can be read over the Smart Energy
 2619 network, but devices that do choose to expose the resource(s) MUST obey the list ordering rules below.

2620 **Table 10-7 Subscription / Notification List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
Subscription	href (ascending)	NA	NA
Notification	href (ascending)	NA	NA

2621 10.6.3 **Application Guidelines / Behavior**2622 10.6.3.1 **Subscription Resource**

2623 A subscription resource is realized as a resource for an individual client, providing interfaces to all
 2624 subscriptions for the given client. A server indicates support of subscriptions for a given resource with
 2625 the subscribable attribute of that resource.

2626 10.6.3.2 **Notification Resource**

2627 The notification resource is used to receive notifications that a resource to which a host is subscribed has
 2628 changed. The location of the notification resource is passed to the subscription server in the body of the
 2629 subscription. As such, a given client (notification resource server) may have one notification resource
 2630 for multiple different notifications or may have a different notification resource for different
 2631 notifications. The resource representation returned in a notification SHALL be identical to that which
 2632 would be returned via a GET request to the resource, subject to the limit parameter discussed below and
 2633 per the rules stated earlier in this specification for representing resources.

2634 While some notification servers (subscription clients) may support reusing a TLS connection as a client
 2635 for notifications, this mechanism is not reliable as a TLS session may have ended. Notification servers
 2636 (subscription clients) SHALL support TLS as a server if they wish to receive secure notifications.

2637 10.6.3.3 **Conditional Subscription / Report by Exception**

2638 If allowed by the specific function set, some resources can have conditional subscriptions to enable a
 2639 report by exception capability. The following conditions are allowed and are specified in the
 2640 subscription:

- 2641 • Lower Threshold – used to cause a notification when the resource's value is below the given
 2642 value.
- 2643 • Upper Threshold – used to cause a notification when the resource's value is above the given
 2644 value.

2645 Note, both an upper threshold and a lower threshold may be specified for a given subscription. If both
 2646 an upper threshold and a lower threshold are specified, the upper threshold SHALL be greater than the
 2647 lower threshold, otherwise an error representation SHALL be returned.

2648 If neither a lower threshold nor an upper threshold is specified, then a server SHALL send a notification
 2649 whenever the resource to which the client is subscribed changes. If a lower threshold and / or an upper
 2650 threshold are specified, then a server SHALL send a notification whenever the appropriate value crosses
 2651 (in either direction) the appropriate threshold for the resource to which the client is subscribed. Servers
 2652 that support subscriptions to a given resource MAY support conditional subscriptions to that resource.

2653 For a given resource, to determine the attribute on which the conditions apply, the attributeIdentifier
 2654 attribute is used.

2655 10.6.3.4 **Subscription Rules**

2656 A subscription renewal is a subscription to the same resource from the same client, regardless of
2657 subscription parameters.

- 2658 1) Clients SHOULD send subscription renewals every 24 hours, and no more often than every 1
2659 hour, to sustain a subscription.
- 2660 2) Servers MAY remove subscriptions at any time.
- 2661 3) Servers SHOULD remove subscriptions if a client has not renewed in 36 hours.
- 2662 4) Subscriptions SHALL be maintained on servers through power loss.
- 2663 5) Servers SHALL use the subscription parameters from the latest subscription renewal.
- 2664 6) Clients SHOULD poll after perceived loss of connectivity.
- 2665 7) If the URI of a resource changes, then subscriptions to that resource SHALL be terminated.
- 2666 8) For subscriptions to non-list resources, a notification SHALL be sent whenever the
2667 representation of the non-list resource changes.
- 2668 9) For subscriptions to list resources, a notification SHALL be sent whenever any subordinate
2669 resources are added to or removed from the list, or if the representation of those subordinate
2670 resources change. As a result, if a client is subscribed to both an event and a list containing that
2671 event, the client will receive two notifications should that event change. See Section 6.7 6.7 for
2672 more information.
 - 2673 a. It should be noted that notifications for list resources are an exception where certain list
2674 attributes are included (e.g., all and results) that would normally not be present when a
2675 list resource is provided in a POST.
- 2676 10) To prevent overwhelming network resources, notifications SHOULD be sent to a given client
2677 for a given resource no more than once every 30 seconds. Notifications for conditional
2678 subscriptions SHOULD only be sent once within this time period for a given client for a given
2679 resource and any additional notifications SHOULD NOT be queued. All devices need to be
2680 considerate of network resources.
- 2681 11) Servers implementing the subscription resource SHALL be capable of maintaining a minimum
2682 of 1 subscription and servers implementing the subscription resource SHOULD support at least
2683 1 subscription per device per subscribable resource.
- 2684 12) If a server implementing the subscription resource is unable to accept a subscription, the server
2685 SHALL return an error resource representation indicating the specific error (e.g., element does
2686 not support conditional subscription) with an HTTP response code of 400.
- 2687 13) When a server removes or terminates a subscription, it SHALL send the client a terminate
2688 subscription (except after an error from an undesired subscription, as mentioned in rule 14
2689 below). The server SHALL send a Notification to the client's "Post URI" for the affected
2690 subscription. The Resource contained in the Notification SHALL be a Subscription, containing a
2691 Status element identifying the reason for terminating the subscription. When a subscription is
2692 terminated because the subscribed resource has moved, servers MAY include newResourceURI
2693 in the subscription termination message, indicating the resource's new location.
- 2694 14) If a client receives a notification for an undesired subscription, the client SHALL return an
2695 HTTP 400 error. Upon receipt of such an error, the server SHALL remove the subscription
2696 without notification.
- 2697 15) Servers SHOULD allow only the end device that corresponds to a given EndDevice resource to
2698 modify the subscriptions within that resource.

2699 16) The default recommended policy is that subscription management SHOULD be performed
 2700 using TLS.

2701 **10.6.4 LogEvents**

2702 **Table 10-8 Subscription / Notification LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
SUB_NTFY_FAIL	0x00	SHOULD be issued when there is a failure to successfully send a Notification (after a successful connection).
SUB_CONN_ESTB_FAIL	0x01	SHOULD be issued when there is a failure to successfully establish a connection to send a Notification.

2703 **10.7 Response**

2704 **10.7.1 Overview**

2705 This function set provides an interface for capturing Responses from all events, including Demand
 2706 Response (DRLC), Distributed Energy Resources (DER), Pricing, and Messaging. Client devices of this
 2707 function set include Smart Thermostats, IHDs, Energy management systems and devices that support
 2708 load control, pricing or text events.

2709 The originating event will indicate if a response is required.

2710 **10.7.2 List Ordering**

2711 **Table 10-9 Response List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
ResponseSet	mRID (descending)	NA	NA
Response	createdDateTime (descending)	endDeviceLFDI (ascending)	NA

2712 **10.7.3 Application Guidelines / Behavior**

2713 The typical use for this function set is that end devices will POST their responses to a Response resource
 2714 identified in events. It is not expected that devices will use discovery to locate these resources.

2715 It is up to the implementer to decide whether to have all responses in a single list or to have multiple
 2716 lists. The destination of the Responses is controlled by the replyTo attribute of the originating event.

2717 If a response is desired to an event, then the event SHALL provide, in the replyTo field, a URI
 2718 indicating the location of where the responses are to be posted.

2719 The client SHALL POST the responses to the indicated URI based on the rules indicated by the
 2720 responseRequired bitfield of the event.

2721 If a server hosts events that specify a response is required then that server is NOT REQUIRED to host
 2722 the response server.

2723 The Response Server identified in the replyTo field of the event SHOULD be available on the network.

2724 Several function sets use the responseRequired attribute, which indicates whether or not this particular
 2725 event requires a response from the client. The following table indicates the valid Response.type values
 2726 supported by each function set:

2727 **Table 10-10 Response Types by Function Set.**

Enumeration Value	Description	Why Sent	When Sent	<i>responseRequired</i> bit position	DRLC and DER	Pricing	Messaging
0	<i>Reserved</i>						
1	Event received	To notify response server that client has initially received event.	When the device first receives the event, either via a GET or a Notification.	0	X	X	X
2	Event started	To notify response server that client has begun event.	At <i>EffectiveStartTime</i> (see Section 12.1.3.1)	1	X	X	X
3	Event completed	To notify response server that the event fully and successfully completed.	At <i>EffectiveEndTime</i> (see Section 12.1.3.1)	1	X	X	X
4	User has chosen to opt-out	To notify response server that user has chosen to opt out of the event. Can occur before event begins.	At time user actively chose to opt out or when device automatically opts out due to user preference.	1	X		
5	User has chosen to opt-in	To notify response server that user has chosen to opt in to the event (after a previous opt-out). Can occur before event begins.	At time user actively chose to opt in or when device automatically opts in due to user preference.	1	X		
6	The event has been cancelled	To notify response server that client has initially received cancellation.	When the device first receives the cancellation, either via a GET or a Notification, even if the event has not yet begun execution.	1	X	X	X

Enumeration Value	Description	Why Sent	When Sent	<i>responseRequired</i> bit position	DRLC and DER	Pricing	Messaging
7	The event has been superseded	To notify response server that client has learned of an event that supersedes this event.	When the device first learns of an event that supersedes this event, even if the event has not yet begun execution.	1	X	X	X
8	Event partially completed with user opt-out	To notify response server that some participation in the event occurred, but not complete participation, due to one or more user opt-outs.	At <i>EffectiveEndTime</i> (see Section 12.1.3.1)	1	X		
9	Event partially completed due to user opt-in	To notify response server that some participation in the event occurred, but not complete participation, due to one or more user opt-ins (after a previous opt-out).	At <i>EffectiveEndTime</i> (see Section 12.1.3.1)	1	X		
10	Event completed, no user participation (previous opt-out)	To notify response server that no participation in the event occurred.	At <i>EffectiveEndTime</i> (see Section 12.1.3.1)	1	X		
11	User has acknowledged the event	To notify response server that the client displayed the event and a human user actively acknowledged it.	At time user actively chose to acknowledge the event.	2	X	X	X
12	Cannot be displayed	To notify response server that the client is unable to display the message.	When the device first receives the event, either via a GET or a Notification.	1			X

Enumeration Value	Description	Why Sent	When Sent	<i>responseRequired</i> bit position	DRLC and DER	Pricing	Messaging
13	Event aborted due to alternate provider event	To notify response server that the client has chosen to stop execution of the event and instead execute an event from a different service provider.	At time the original event was ceased.	1	X	X	X
14 - 251	<i>Reserved</i>						
252	Rejected – control parameter not applicable	To notify response server that client is unable to execute the event due to the controls being inapplicable to the device (e.g., a temperature offset was sent to a pool pump).	When the device first receives the event, either via a GET or a notification.	1	X		
253	Rejected – invalid event	To notify response server that client is unable to execute the event due to the controls (e.g., duty cycle, offset, setpoint) being out of range or not possible for the device (e.g., settings already at a maximum or minimum).	When the device first receives the event, either via a GET or a notification.	1	X		
254	Rejected – event was received after it expired	To notify response server that client has initially received event, but that the event was received after <i>SpecifiedEndTime</i> .	When the device first receives the event, either via a GET or a Notification.	1	X	X	X
255	<i>Reserved</i>						

2728 10.7.3.1 **Response Storage**

2729 It is expected that the server provides some mechanism that allows the service provider to obtain the
 2730 responses, even in the presence of outages. Details of storage are vendor specific.

2731 Implementers SHOULD have enough storage and bandwidth to support responses for the number of
2732 devices they plan to support as a server for each function set that requires a response.

2733 If the server supports the GET method for the response function set it SHALL minimally support 1
2734 response for each function set for which it accepts responses.

2735 **10.7.4 LogEvents**

2736 **Table 10-11 Response LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
RESPONSE_LIST_OVERFLOW	0x00	Should be issued by a server anytime their response list has exceeded maximum storage.
RESPONSES_REPORTED_UPSTREAM	0x01	Should be issued by a server when responses are sent upstream.

2737

11 Common Resources

2738 This section defines the resources and function sets that provide general purpose, non-domain specific
2739 functionality.

2740 **11.1 Time Function Set**

2741 **11.1.1 Overview**

2742 Devices synchronize their time source by locating and acquiring time from a Time resource. Time
2743 synchronization of devices is required to effectively support DRLC, pricing changes, time stamping of
2744 metering data, etc.

2745 Devices implementing the Time resource obtain correct time either directly from an external
2746 authoritative time source (e.g., NTP) or indirectly from an inaccurate source (e.g., manually entered via
2747 UI).

2748 There may be a dependency upon an external time source for UTC time (NTP service, etc.).

2749 **11.1.2 List Ordering**

2750 This resource does not contain any lists so no ordering is specified.

2751 **11.1.3 Application Guidelines / Behavior**

2752 All devices SHALL implement the Time function set as a resource server, or a client, or both.

2753 All communication of time used throughout the specification, with the exception of time for user
2754 display, SHALL be according to the definition of TimeType. Devices with user displays SHALL
2755 support local time and daylight savings time offsets.

2756 If FunctionSetAssignments contain both Event-based function sets (e.g., DRLC, pricing, message) and a
2757 Time resource then devices SHALL use the Time resource from the same FunctionSetAssignments
2758 when executing the events from the associated Event-based function set.

2759 For example, if there are two FunctionSetAssignments 'A' and 'B', where 'A' contains a DRLC function
2760 set (DRLC(A)) and a Time resource (Time(A)) and 'B' contains a DRLC function set (DRLC(B)) and a
2761 Time resource (Time(B)), then events from DRLC(A) will be executed using Time(A) as their Time
2762 resource and events from DRLC(B) will be executed using Time(B) as their Time resource, regardless
2763 of the quality metrics of either Time resource.

2764 If a device is not assigned a Time resource, it MAY discover and operate Event-based function sets.
2765 When the device executes these events, the device SHALL use the Time resource from the server device
2766 from which it received the event. If a client discovers an Event-based function set and cannot also
2767 retrieve a Time resource from this same server, it SHALL NOT act on the events from this Event-based
2768 function set.

2769 If a device is not processing events it SHALL discover and choose the Time resource with the best
2770 quality metric.

2771 If a device displays time then it SHOULD display the time with the highest quality metric from one of
2772 the above-acquired Time resources. For devices displaying time, it should be noted that the Time
2773 resource with the highest quality metric may differ from the Time resource(s) used by events (e.g., a
2774 service provider's Time resource may "true-up" at the end of a billing cycle). Devices SHOULD either
2775 convert displayed event times to be consistent with the chosen Time resource or make it clear to users
2776 that there are Time resource differences.

2777 Some service providers operate a given asset on a time standard that differs from the definition given for
2778 TimeType. This would be an intentionally uncoordinated time. If a time server serves an intentionally

2779 uncoordinated time in the currentTime field, it SHALL have a quality metric of 7 - time intentionally
 2780 uncoordinated.

2781 See the XML schema described in [ZB 13-0201] for details of the quality metric.

2782 Devices SHOULD periodically query the selected Time resources and compare the response with the
 2783 local device time to determine local clock accuracy.

2784 A device SHOULD manage the frequency of its Time resource updates and subsequent local clock
 2785 updates to provide local time accurate to within:

2786 1) 10 seconds per 24 hour period for devices displaying time to the user.

2787 2) 60 seconds per 24 hour period if there is no user interface.

2788 To maintain time synchronization, devices SHOULD maintain these accuracies via Time resource
 2789 requests. To keep network traffic to a minimum this polling SHALL be no more than once per 15
 2790 minutes once a valid time source has been established. Requests SHALL NOT exceed once per 15
 2791 seconds and SHALL employ exponential back off prior to establishing a valid time source. These
 2792 requirements apply on a per time server basis.

2793 Time related configuration is handled within the Configuration resource.

2794 Time adjustments less than 60 seconds SHALL never be made backwards (e.g., use stall time or long
 2795 seconds to correct for being ahead on time).

2796 11.1.4 **LogEvents**

2797 **Table 11-1 Time LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
TM_TIME_ADJUSTED	0x00	Should be issued by a server when its time is adjusted.
TM_TIME_SOURCE_CHANGED	0x01	Should be issued by a client when it chooses a different primary time source.

2798 **11.2 DeviceInformation Function Set**

2799 **11.2.1 Overview**

2800 The DeviceInformation function set provides static manufacturer specific information about a device.

2801 **11.2.2 List Ordering**

2802 This resource does not contain any lists so no ordering is specified.

2803 **11.2.3 Application Guidelines / Behavior**

2804 The DeviceInformation function set is used to provide information about the actual device. Any device
 2805 that implements a HTTP server to serve resources SHALL implement the DeviceInformation resource
 2806 (this does not apply to devices that only serve Notification / NotificationList resources for the purpose of
 2807 receiving subscription notifications). Client devices that do not implement a HTTP server are not
 2808 required to serve the DeviceInformation function set. However this information MAY be posted to the
 2809 appropriate EndDevice resource on an EndDevice function set server.

2810 **11.2.4 LogEvents**

2811 There are no LogEvents generated by this function set.

2812 11.3 **Power Status**2813 11.3.1 **Overview**

2814 The PowerStatus resource provides information regarding a device's current power source, as well as
 2815 basic status regarding any battery installed within the device.

2816 11.3.2 **List Ordering**

2817 This resource does not contain any lists so no ordering is specified.

2818 11.3.3 **Application Guidelines / Behavior**

2819 The content of this resource will change as the device switches power sources (mains to battery, battery
 2820 to mains) and as the battery charges / discharges.

2821 11.3.4 **LogEvents**2822 **Table 11-2 Power Status LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
PS_LOW_BATTERY	0x00	Should be issued when the installed battery charge drops below the configured low battery charge threshold.

2823 11.4 **Network Status**2824 11.4.1 **Overview**

2825 The Network Status function set provides information regarding the device's network (IP) layer, and
 2826 potentially link layer, performance.

2827 The design of this function set is intended to allow a single network (IP) layer interface to be linked with
 2828 multiple link layer interfaces (as is often the case with devices that provide L2 bridging) as well as to
 2829 allow multiple network (IP) layer interfaces to be linked with a single link layer interface.

2830 Several references were consulted in the generation of the design and attributes of this function set,
 2831 including: [RFC 2863], [RFC 4293], [RFC 3635], [DOCSIS], and [Linux].

2832 The resources of this function set can also serve as an example for others to create additional network
 2833 status information in a separate namespace (e.g., for other link layers).

2834 11.4.2 **List Ordering**2835 **Table 11-3 Network Status List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
IPInterface	ifIndex (ascending)	NA	NA
IPAddr	address (ascending)	NA	NA
LLInterface	EUI64 (ascending)	NA	NA
RPLInstance	DODAGid (ascending)	RPLInstanceID (ascending)	NA

Resource Name	Primary Key	Secondary Key	Tertiary Key
RPLSourceRoutes	SourceRoute (ascending)	DestAddress (ascending)	NA

2836

2837 11.4.3 **Application Guidelines / Behavior**2838 This function set does not have any specific application guidelines / behavior beyond those already
2839 specified in the schema.2840 11.4.4 **LogEvents**2841 **Table 11-4 Network Status LogEvents**

LogEvent Name	LogEvent Code	LogEvent Description
NS_JOIN_UNSP	0x00	SHOULD be issued when the device joins an unspecified network
NS_LVE_UNSP	0x01	SHOULD be issued when the device leaves an unspecified network
NS_JOIN_802_3	0x02	SHOULD be issued when the device joins an IEEE 802.3 (Ethernet) network
NS_LVE_802_3	0x03	SHOULD be issued when the device leaves an IEEE 802.3 (Ethernet) network
NS_JOIN_802_11	0x04	SHOULD be issued when the device joins an IEEE 802.11 (WLAN) network
NS_LVE_802_11	0x05	SHOULD be issued when the device leaves an IEEE 802.11 (WLAN) network
NS_JOIN_802_15	0x06	SHOULD be issued when the device joins an IEEE 802.15 (PAN) network
NS_LVE_802_15	0x07	SHOULD be issued when the device leaves an IEEE 802.15 (PAN) network
NS_JOIN_1901	0x08	SHOULD be issued when the device joins an IEEE 1901 (PLC) network
NS_LVE_1901	0x09	SHOULD be issued when the device leaves an IEEE 1901 (PLC) network

2842 11.5 **LogEvent List**2843 11.5.1 **Overview**2844 The LogEvent List contains a list of time-stamped instances of LogEvents generated by the device.
2845 LogEvents are typically asynchronously generated warnings of some out of bounds condition or unusual
2846 significant event detected by the device. For example, the device's battery charge falling below an
2847 operator designated low charge threshold.2848 11.5.2 **List Ordering**2849 **Table 11-5 LogEvent List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
---------------	-------------	---------------	--------------

Resource Name	Primary Key	Secondary Key	Tertiary Key
LogEvent	createdDateTime (descending)	LogEventID (descending)	NA

2850

2851 11.5.3 **Application Guidelines / Behavior**

2852 The number of LogEvents supported within the LogEvent List SHALL be vendor dependent. LogEvent
 2853 server devices SHALL be capable of internally storing and supporting at least 10 unique LogEvent
 2854 instances. When a new LogEvent is issued, it SHALL be added as the first entry in the LogEvent List. If
 2855 the log is full, the oldest LogEvent SHOULD be removed to make room for the latest incoming
 2856 LogEvent.

2857 LogEvent List is OPTIONAL for both servers and clients.

2858 LogEvents SHALL be time stamped when they are added to the LogEventList. A server device therefore
 2859 needs a time source, and should be using the highest quality Time resource it can obtain from the HAN.

2860 A LogEvent SHALL be ZigBee Alliance defined or vendor defined or defined within other profiles.

2861 11.5.3.1 **LogEvents**

2862 There are no LogEvents generated by this function set.

2863 The table below presents the LogEvent codes allocated by the ZigBee Alliance for general Smart Energy
 2864 2 events. These codes are associated with the value of zero (General) in the functionSet field.

2865 **Table 11-6 General LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
LE_GEN_HARDWARE	0x00	Unclassified general hardware fault occurred.
LE_GEN_SOFTWARE	0x01	Unclassified general software fault occurred.

2866 Note: See the function set definitions for LogEvent Codes associated with each function set.

2867 11.6 **Configuration Resource**2868 11.6.1 **Overview**

2869 The Configuration resource implements centralized read / write access to the device's operational
 2870 configuration. This resource may be queried to provide current device configuration, perform
 2871 configuration backup, etc. This resource is modified as needed to control the operation of the device.

2872 This resource adheres to the recommendations outlined in [RFC 3535]. Specifically, configuration is
 2873 best separated from operational state and performance statistics to ease configuration retrieval,
 2874 conformance check, and placement / modification to the device. In other words, it is easier to manage
 2875 device configuration via a single resource than having to collect, collate, and update configuration
 2876 parameters from multiple resources spread across a SEP 2 implementation.

2877 11.6.2 **List Ordering**

2878 This resource does not contain any lists so no ordering is specified.

2879 11.6.3 **Application Guidelines / Behavior**

2880 It is strongly RECOMMENDED that devices persist configuration data across a power cycle.

2881 11.6.4 **LogEvents**2882 **Table 11-7 Configuration LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
CFG_CONFIGURATION_UPDATED	0x00	SHALL be issued when the local Configuration resource is changed.

2883 11.7 **File Download Function Set**

2884 This section describes the mechanisms used to support secure, interoperable, remote software download
 2885 to Smart Energy Profile 2.0 devices. This function set may also be used for remote download of other
 2886 file artifacts such as log files, configuration files, security credential files, etc. Three resources constitute
 2887 the File Download Function Set: the FileList resource, the File resource, and the FileStatus resource.

2888 There are two primary actors involved in a file download: the Loading Device (LD) and the File Server
 2889 (FS). A FS is any device serving this function set.

2890 To accommodate sleepy end devices, SEP 2 supports only a polled mode of file download.

2891 11.7.1 **File List Resource**2892 11.7.1.1 **Overview**

2893 A FileList resource contains zero or more File resources available to be loaded by client devices. A File
 2894 resource contains various meta data describing the file and a link to the file binary artifact.

2895 11.7.1.2 **List Ordering**

2896 File resources are ordered within a FileList as follows:

2897 **Table 11-8 File Download List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key	Quaternary Key
File	mfID (ascending)	mfModel (ascending)	mfVer (descending)	href (ascending)

2898 The main use case for this ordering is to enable client devices (of specific manufacturer and model) to
 2899 query the FileList for available file types with a version newer than the file currently on the device.

2900 11.7.1.3 **Application Guidelines / Behavior**

2901 This specification designates files resident on the LD as activated or non-activated. A non-activated file
 2902 is a file that has been physically loaded onto the LD, but not yet placed into operation. An activated file
 2903 is a file that has been both physically loaded onto the LD and placed into operation. An example use of
 2904 the activation state is the scenario in which an operator first must load and confirm load of new software
 2905 images to a large set of devices (as non-activated, not running) before setting that software image to an
 2906 activated state (the running image).

2907 A FileList MUST contain File resources for each file made available by the FS for download by LDs.
 2908 The FileList MAY be discoverable as specific to a device (FunctionSetAssignment accessed via an
 2909 EndDevice) and / or MAY be discoverable to all devices via DeviceCapabilities.

2910 The LD issues discovery requests to locate available FS (DNS-SD subtype "file"). The LD MUST first
 2911 determine if there exists a FunctionSetAssignment, containing a FileList, for the device. If such a
 2912 FileList exists, the LD MUST use this FileList exclusively. If a Time resource is specified within this
 2913 same FunctionSetAssignment, the LD MUST use this Time resource as the reference for file activation

2914 time. If the LD does not locate a FunctionSetAssignment directing the LD to a specific FileList, the LD
2915 MUST attempt discovery of a FileList provided by any device's DeviceCapabilities resource.

2916 If the LD locates more than one File resource satisfying its File query, it is up to the LD to determine the
2917 appropriate File selection. Any algorithm used to make this selection is out of scope of this
2918 specification.

2919 If the LD is unable to make a selection between multiple files, the LD SHOULD issue an
2920 FE_FILE_MULTIPLE_FILES LogEvent.

2921 An LD MUST poll for available files at least once every 24 hours.

2922 File content is transferred using either HTTP or HTTPS. The latter SHOULD be used when encryption
2923 over the air / wire is desired. The choice is left to the manufacturer. An LD SHALL provide the ability
2924 to fully receive and store a non-activated file while the current activated file remains operational. For
2925 example, an LD SHALL be able to load a new software image (a non-activated file) while the current
2926 software image executes. In the case of a software image, the activated file is the running image.

2927 SEP 2 deployments could contain clients and servers of vastly differing capabilities: from constrained
2928 embedded devices to cloud-based services. It is thus important that FSs and LDs be provided ways to
2929 control the rate at which files are transferred. The HTTP Range and Content-Range entity headers are
2930 used to support incremental loading of large files. LDs SHOULD use the Range entity header, within
2931 HTTP GET requests for file content, to support file loading via a sequence of one or more HTTP
2932 requests. An FS MUST be able to process the Range entity header within HTTP GET requests for file
2933 content, and MUST support Range entity header processing for at least a single range of bytes. An FS
2934 MUST include the Content-Range entity header within HTTP responses for file content when an LD
2935 requests a specific range. An LD SHOULD be able to process Content-Range entity headers within
2936 HTTP responses for file content.

2937 The HTTP Etag entity header is used to allow LDs to detect any modification to a resource. This is of
2938 particular importance to detect changes in file content during a long running (multi-GET) file load. An
2939 FS MUST maintain a unique entity-tag value for each version of a file referenced by a specific URI. Per
2940 [RFC 2616], the entity-tag value MUST change if the underlying bits of the file change. The FS MUST
2941 include an Etag entity header (indicating the file entity-tag value) in all GET responses containing file
2942 content. An LD MUST detect a change in file Etag value. An LD MUST abort the file load if the Etag
2943 has changed, and SHOULD restart the file load (now based on the new Etag). Note that this approach
2944 for file modification detection was decided, in the general case, to be more bandwidth efficient than
2945 forcing the LD to include an If-Match or If-Range entity header on all GET requests for file content.

2946 For previously loaded, non activated files for which an activation time has not been acquired, the LD
2947 MUST poll the File resource at least once every 24 hours for inclusion of a file activation time. The LD
2948 SHOULD cease attempting to acquire activation time after 5 unsuccessful attempts or if a new File of
2949 same type has been loaded (whichever condition occurs first). If the LD is unable to acquire an
2950 activation time after 5 attempts, the LD MUST cease any further attempts and consider the activation
2951 failed.

2952 An FS SHOULD maintain internal estimates of its current resource usage. If an FS determines that it is
2953 unable to service an incoming HTTP request, the FS SHOULD issue a response indicating 503 error and
2954 SHOULD include the Retry-After entity header (providing guidance to the LD for retry attempt). The
2955 Retry-After entity header SHOULD provide a best estimate as to when the FS expects it will be able to
2956 service the LD request.

2957 An LD SHOULD implement handling for 503 errors and SHOULD implement handling of the Retry-
2958 After entity header. If the LD does not support processing of the Retry-After entity header, the LD
2959 MUST wait at least 30 seconds before retrying the file content request.

2960 An LD SHOULD negotiate a desired TCP window size with the FS.
2961 Files SHOULD be treated as raw binary, thus a Content-Type entity header of "application/octet-stream"
2962 SHOULD be used.

2963 11.7.1.3.1 **File Formats**

2964 SEP 2 makes no requirements upon the internal structure of files. A file is simply a manufacturer-
2965 defined sequence of binary octets whose internal details are completely defined by the manufacturer and
2966 out of scope of this specification.

2967 SEP 2 does, however, require a file be digitally signed to protect the file from undetected modification
2968 and provide file origin authentication.

2969 11.7.1.3.2 **File Signatures**

2970 Files SHALL be signed and signatures verified using approved algorithms specified in [NIST SP800-
2971 131A], [NIST SP800-131B], and [NIST SP800-131C]. Cryptographic strength MUST be at least the
2972 minimum specified in [ZB 09-5449] Section 11.3 (which is 128-bit). This specification does not
2973 otherwise prescribe specific cryptographic mechanisms via which the file is signed or signature verified,
2974 does not prescribe the format of the file signing trust chain, and does not prescribe how that trust chain
2975 is managed on the LD. There is no required relationship between the CSEP device trust chain and the
2976 file signing trust chain. The signature of the file SHALL be calculated over the entire contents of the file
2977 excluding the signature octets.

2978 11.7.1.3.3 **Loading Files Containing Security Credentials**

2979 File loads containing SEP 2 security artifacts (type = 1 or any manufacturer defined file type containing
2980 credentials, key material, etc.) MUST be protected to at least the current level of security associated with
2981 the artifact being loaded. Such a file load MUST be secured using an HTTPS connection.

2982 11.7.1.3.4 **File Query Parameters**

2983 Beyond the standard SEP 2 list query parameters for result start / length / etc., a FileList supports
2984 additional query parameters to support filtering of Files result sets. These file query parameters,
2985 modeled after several of the elements within the File resource, are:

- 2986 1) type
- 2987 2) lFDI
- 2988 3) mfHwVer
- 2989 4) mfID
- 2990 5) mfModel
- 2991 6) mfSerNum
- 2992 7) mfVer

2993 All of the query parameter values MUST adhere to identical syntax and semantics as used for the
2994 correspondingly named elements of the File resource described in [ZB 13-0201].

2995 An FS MUST be able to process all file query parameters. LD usage of any of the file query parameters
2996 is OPTIONAL.

2997 File query parameters MUST be applied to a File List before the standard list query parameters (i.e., the
2998 standard query parameters are used to page the list resulting after the file query parameters are applied).

2999 The set of file query parameters MUST be interpreted as a Boolean expression, with each file query
3000 parameter considered a clause of that expression, and each clause connected by a logical AND operand.

3001 Each file query parameter MUST provide an exact match with its File equivalent for its clause to
 3002 evaluate to true. There is one exception to this case: the mfVer MUST be evaluated as a GREATER
 3003 THAN logical operand versus its File equivalent.

3004 Omission of a query parameter MUST be interpreted as a match, regardless the File's corresponding
 3005 value.

3006 11.7.1.4 **LogEvents**

3007 Loading and activation of files is a long running operation. Diagnosing problems can be challenging.
 3008 LDs SHOULD implement the LogEvent Log. If an LD does implement the LogEvents for this function
 3009 set, the LD SHALL implement all of the following LogEvents:

3010 **Table 11-9 File Download LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
FE_FILE_LOAD_FAILED	0x00	This event SHALL be recorded by the LD when it is unable to load a file that was indicated as available by the FS.
FE_FILE_LOAD_OK	0x01	This event SHALL be recorded by the LD when a file is successfully loaded to the LD.
FE_FILE_SIGNATURE_VERIFICATION_FAILED	0x02	This event SHALL be recorded by the LD when it is unable to verify the signature of a loaded file.
FE_FILE_SIGNATURE_VERIFICATION_OK	0x03	This event SHALL be recorded by the LD when a loaded file's signature is successfully verified.
FE_FILE_ACTIVATION_FAILED	0x04	This event SHALL be recorded by the LD when it is unable to activate a previously loaded and signature verified file.
FE_FILE_ACTIVATION_OK	0x05	This event SHALL be recorded when the LD has successfully activated a loaded and signature verified file.
FE_FILE_MULTIPLE_FILES	0x06	This event SHALL be recorded when the LD is unable to determine a single file selection from a set of 2 or more files it has discovered.

3011 11.7.2 **File Status Resource**

3012 11.7.2.1 **Overview**

3013 The FileStatus resource enables LDs to provide a means to track the progress of any file load operations
 3014 and also a means to diagnose failed file load operations.

3015 11.7.2.2 **List Ordering**

3016 This resource does not contain any lists so no ordering is specified.

3017 11.7.2.3 **Application Guidelines / Behavior**

3018 An LD SHOULD implement a SelfFileStatus resource or an LD SHOULD support PUT to a remote
3019 FileStatus (on an EndDevice server). The SelfFileStatus and FileStatus resources SHOULD be updated
3020 whenever a status transition occurs (as a File load and activation flow executes). Status is discussed in
3021 the definition of FileStatus (see [ZB 13-0201]).

3022 11.7.2.4 **LogEvents**

3023 There are no LogEvents generated by this resource.

3024 12 Smart Energy Resources

3025 This section defines the function sets that are specific to the domain of Smart Energy. The section
3026 begins with a sub-section defining behaviors and discussing issues that are common to several of the
3027 following function set definitions.

3028 12.1 Common Functionality

3029 12.1.1 Overview

3030 This Section describes common application functionality guidelines for features like randomization and
3031 active URIs that apply to more than one function set or in circumstances like adjacent or superseding
3032 events where special cases arise or need clear guidance to result in predictable behavior.

3033 12.1.2 Active List Elements

3034 The Active List Elements of a function set allow client devices to easily find the active instance of a
3035 specific resource. This makes it easier for constrained devices to find the information they are looking
3036 for without the need to explore a large list. Many of the function sets will provide this resource,
3037 including Demand Response, Messaging, and Pricing.

3038 Active lists contain a subset of the instances in the full list of events for a given resource.

3039 12.1.3 Event Rules and Guidelines

3040 These rules and guidelines are meant to specify client behavior in situations in which adjacent, nested,
3041 or overlapping events could cause clients to behave in an unexpected or undesirable manner.

3042 12.1.3.1 Definitions

3043 The following definitions define terms that will be used throughout the Event Rules and Guidelines
3044 section. These terms, when used, are *italicized* for emphasis.

3045 **Event** – refers to any instance of a resource with a defined valid duration for which a client should take
3046 action. This would include all resources that are derived from the Event resource defined in the schema
3047 [ZB 13-0201].

3048 **Start Time** – Contained within the interval attribute of an *Event* resource representation and indicates
3049 when the *Event* should start.

3050 **Duration** – Contained within the *interval* attribute of an *Event* resource representation and indicates for
3051 how long the *Event* is active.

3052 **Specified End Time** – Time when an *Event* completes as specified in the instance's interval attribute.
3053 This is calculated by adding *Duration* to *Start Time*.

3054 **Scheduled Period** - Represents the time between the *Start Time* and the *Specified End Time* of the
3055 *Event*. A Scheduled Period is a Duration anchored at a specific *Start Time*.

3056 **Effective Start Time** - Represents the time at which an *Event*'s interval attribute indicates
3057 commencement based on the *Start Time* plus or minus any applied Start Randomization offsets as
3058 calculated by the Client.

3059 **Earliest Effective Start Time** - Represents the earliest time at which an *Event*'s interval attribute could
3060 indicate commencement. Calculated as the minimum of *Start Time* or *Start Time* plus the Start
3061 Randomization specified by the event.

3062 **Effective End Time** - Represents the time at which an *Event*'s interval attribute indicates completion
3063 based on the *Effective Start Time*, plus *Duration*, plus or minus any applied Duration Randomization
3064 offsets as calculated by the Client.

- 3065 **Effective Scheduled Period** - Represents the time between the *Effective Start Time* and the
 3066 *Effective End Time*.
- 3067 **Duration Randomization** – The bound on the amount of time to be used when randomizing the
 3068 completion of an *Event*.
- 3069 **Effective Duration** - Represents *Duration* with *Duration Randomization* applied.
- 3070 **Overlapping Event** - Defined as an *Event* where the *Scheduled Period* covers part or all of an existing,
 3071 previously scheduled *Event*.
- 3072 **Start Randomization** – The bound on the amount of time to be used when randomizing the
 3073 commencement of an *Event*.
- 3074 **Successive Events** - When the *Start Time* plus *Duration* of the first event is the same as the
 3075 *Start Time* of the second event without randomization.
- 3076 **Nested Events** - Defined as two *Events* where the scheduled *Start Time* and *Specified End Time* of the
 3077 second *Event* falls during the *Scheduled Period* of the first *Scheduled Event* and the second *Event* is of
 3078 shorter *Duration* than the first *Event*. This is an extreme case of *Over Lapping Event*.

3079 12.1.3.2 **Rules and Guidelines**

3080 In the text below, some rules and guidelines will be identified as specific to "function sets with direct
 3081 control." These function sets exert direct control over clients' behavior and are typically used to manage
 3082 service provider infrastructure through incentive programs to the HAN owner. This is in contrast with
 3083 informational function sets that provide data to clients for display and awareness purposes such as
 3084 Billing, Messaging, and Pricing. Function sets with direct control primarily include Flow Reservation,
 3085 DRLC and DER. Pricing is somewhat unique in that it exerts control-like effects upon price responsive
 3086 clients; however, it is not a form of direct control from the service provider's point of view. The
 3087 following rules and guidelines apply to both types of function sets unless specifically noted.

3088 The depicted behaviors and required application management decisions are driven from the following
 3089 guidance and rule set:

- 3090 1) Clients that act on events that do not subscribe to their *Event* lists SHALL poll the lists for new
 Events at least once every 15 minutes and SHOULD poll at least every 5 minutes.
- 3091 2) Clients SHALL monitor the active *Event(s)* for status changes at least once every 15 minutes
 and SHOULD monitor at least once every 5 minutes. Note that these resources might be
 acquired in meeting requirement 1 above; additional polling might not be needed.
- 3092 3) Editing *Events* SHALL NOT be allowed except for updating status. Service providers SHALL
 cancel *Events* that they wish clients to not act upon and / or provide new superseding *Events*.
- 3093 4) For function sets with direct control and the Pricing function set, clients SHALL NOT
 simultaneously execute or report execution of multiple simultaneous *Events*. (e.g., *Nested*
 Events and *Overlapping Events*). The rules below clarify the expected behavior in cases in
 which either of these situations arises.
- 3094 5) A client SHALL consider the current *Event* complete if a superseding *Event* is started.
- 3095 6) When comparing two *Nested Events* or *Overlapping Events*, from servers with the same primacy
 the creationTime element SHALL be used to determine which *Event* is newer and therefore
 supersedes the older. The *Event* with the larger (e.g., more recent) creationTimedate-time is the
 newer *Event*.

- 3108 7) *Events* presented to the HAN SHOULD make minimum use of *Overlapping Events* and
3109 *Nested Events*. *Overlapping Events* and *Nested Events* SHOULD only be used where changing
3110 conditions mandate superseding previous *Events*.
- 3111
- 3112 8) When changing conditions mandate changes in the sequence or contents of *Events*, the
3113 following guidelines MAY be used to indicate desired actions:
3114 a. Canceling existing *Events* and reissuing new *Events*.
3115 b. Sending overlapping or nested *Events* to supersede existing *Events*.
- 3116
- 3117 9) When a *Nested Event* completes the containing / superseded *Event* SHALL NOT be reinstated
3118 and remain in a superseded state.
- 3119
- 3120 10) For function sets with direct control, it is RECOMMENDED that process 8.b be used for most
3121 situations since it can allow a smoother change between two sets of directives but in no way
3122 does it negate the responsibilities identified in rule #7.
- 3123
- 3124 11) Clients SHALL verify the EventStatus of an *Event* before acting upon it. If the EventStatus
3125 potentiallySupercededTime has changed since last checked, and if the EventStatus type is
3126 "Partially Superseded", clients SHALL check all *Events* from that function set instance that may
3127 supersede the original *Event*.
- 3128
- 3129 12) When a client receives an *Event* with the *Specified End Time* in the past (*Specified End Time* <
3130 Current Time), this *Event* SHALL be ignored. Note that the *Duration Randomization* is not used
3131 in this calculation.
3132 a. For function sets with direct control, if the *Event* responseRequired indicates, clients
3133 SHALL POST a Response to the replyTo URI with a Status of "Rejected - *Event* was
3134 received after it had expired".
- 3135
- 3136 13) When a client receives an *Event* and calculates an *Effective Start Time* (*Start Time* + *Start*
3137 *Randomization*) in the past and a *Specified End Time* in the future ((*Effective Start Time* <
3138 Current Time) AND (*Specified End Time* > Current Time)), the client SHALL begin the *Event*
3139 using the current time and the absolute value of *Start Randomization*. For response reporting
3140 purposes, the start time SHALL be reported as the Current Time plus applied *Start*
3141 *Randomization* applied. For *Event* duration purposes, the *Specified End Time* SHALL be
3142 preserved, and any *Duration Randomization* attributes SHALL be applied to the abbreviated
3143 *Duration*.
- 3144
- 3145 14) For function sets with direct control, regardless of the state of an *Event* (scheduled or active),
3146 when a client detects an *Overlapping Event* condition, the *Event* with the latest creation time
3147 will take precedence over the previous *Event*. Depending on the state of the *Event* (scheduled or
3148 active), one of the following steps SHALL take place:
3149 a. If the previous *Event* is scheduled and not active and if the *Event* responseRequired
3150 indicates, the client SHALL POST a Response (referencing the previous *Event*) with the
3151 Status of "The *Event* has been superseded." After the Response has been successfully
3152 POSTed, the client SHALL ignore the previous *Event* scheduled.
3153 b. If the previous *Event* is active, the client SHALL change directly from its current state
3154 to the requested state at the effective start time of the *Overlapping Event*. If the *Event*
3155 responseRequired indicates, the client SHALL POST a response (referencing the
3156 previous *Event*) with a Status of 'The event has been superseded' at the effective start
3157 time of the *Overlapping Event*.

3158

3159 15) Randomization SHALL NOT cause *Event* conflicts or unmanaged gaps. To clarify:

- 3160 a. For *Successive Events* clients SHALL use the earlier *Event*'s Effective *End Time* as the
 3161 *Effective Start Time* of the later *Event*. *Events* are not reported as superseded and Clients
 3162 should report *Event* statuses as they normally would for a set of *Successive Events*.
 3163 Note: This means that a group of *Successive Events* without *Duration Randomization*
 3164 will run successively using the initial *Start Randomization* for each of the *Events* in the
 3165 group.
- 3166 b. Randomization SHALL NOT artificially create a gap between *Successive Events*.

3167

3168 16) It is permissible to have gaps when *Events* are not *Successive Events* or *Overlapping Events*.

3169

3170 17) If multiple EndDeviceCategoryTypes are identified for an *Event*, future *Events* for an individual
 3171 EndDeviceCategoryType (or a subset of the original *Event*) that cause an *Overlapping Event*
 3172 will supersede the original *Event* strictly for that EndDeviceCategoryType (or a subset of the
 3173 original *Event*). Note: Rule #6 applies to all *Overlapping Events*.

3174

- 3175 a. Those clients whose EndDeviceCategoryType is not listed in the future *Event* but whose
 3176 EndDeviceCategoryType was included in the original *Event* SHALL continue to
 3177 execute per the parameters of the original *Event*.
- 3178 b. Rule #3 continues to apply. Servers SHALL NOT edit the original *Event* but SHALL
 3179 maintain all *Events* in their entirety.
- 3180 c. A server SHALL set the potentiallySuperseded flag when the *Event* is superseded for
 3181 any of the device categories and update the potentiallySupersededTime.

3182

3183 18) Servers SHOULD maintain and serve *Events* for the maximum Effective Scheduled Period.
 3184 This applies even if the *Event* in question is cancelled, so as to support devices that may have
 3185 previously received a copy of the *Event* from the server.

3186

3187 19) When an *Event* is removed from the server (e.g., due to limited storage space for the Event list)
 3188 clients SHALL NOT assume the *Event* has been cancelled. Client devices SHALL only act on a
 cancellation as indicated in the rules above or an update to the *Event's Status* attribute.

3189

12.1.4 **Randomization**

3190

12.1.4.1 **Overview**

3191

3192 The primary goal of randomization is to mitigate the effect of simultaneous actions by large groups of
 3193 devices on distribution infrastructure (e.g., in response to a control signal). Adverse effects include
 3194 voltage surges or sags, which can ripple through the distribution infrastructure and cause serious
 3195 reliability problems or damage to electronic devices. Randomization is also useful to the orderly
 3196 shutdown of operations prior to an event taking place to ensure the desired response occurs at the
 3197 desired time. Use of randomization is appropriate and necessary to ensure the reliable and orderly
 3198 operation of commodity distribution infrastructure.

3199

3200 The randomization mechanism is suitable for any function set that signals devices to change behavior or
 3201 causes actions to be taken in response. It is intended as a common object for all function sets. The
 3202 primary function sets that use randomization are DRLC and Pricing. The DER function set may also
 make use of randomization.

3203

12.1.4.2 **Randomization Attributes**

3204

Randomization consists of two signed attributes: *randomizeStart* and *randomizeDuration*. Neither, one,
 or both of these may be included as part of an *Event*.

3205 Randomization is implemented using signed integer(s) to indicate whether a HAN device executes a
3206 control action before or after the start or duration (or both start and duration) of the related *Event*. Time
3207 granularity for randomization is one second.

3208 The randomization values (randomizeStart and randomizeDuration) are generated by the creator of the
3209 *Event*. Clients acting on these *Events* use these values as bounds for generating a local random offset to
3210 the start and duration times, respectively, according to rules given below. Any device handling these
3211 *Events* and not operating on them SHALL NOT modify or apply them.

3212 Examples of how to set the randomization parameters to accomplish different load management
3213 techniques can be found in Section 16.20.

3214 12.1.4.2.1 **randomizeStart**

3215 The randomizeStart attribute represents the bound on the number of seconds to be used when
3216 randomizing the commencement of an *Event*. The device SHALL randomly select a number in seconds
3217 from zero to the randomizeStart specified for this event. Depending on the sign of the value (positive or
3218 negative), randomization SHALL be applied before or after the scheduled *Start Time* of the *Event*. If the
3219 value is negative, randomization SHALL be applied before the specified *Start Time* of the *Event*.

3220 For example, if an *Event* is scheduled to start at 11:00AM and has a *randomizeStart* value of "-300"
3221 (e.g., five minutes prior randomization), a client could randomly generate a number of "-120" (e.g., two
3222 minutes prior randomization) and begin the *Event* at 10:58AM. If the value is positive, randomization
3223 SHALL be applied after the specified *Start Time* of the event, causing the device to delay
3224 commencement of the *Event*.

3225 12.1.4.2.2 **randomizeDuration**

3226 The randomizeDuration represents the bound on the number of seconds to be used when randomizing
3227 the Duration of an *Event*. Devices SHALL randomly select a number in seconds from zero to the
3228 randomizeDuration value specified for the *Event*. Depending on the sign of the value (positive or
3229 negative), randomization SHALL be applied to increase or decrease the Duration of the *Event*. If the
3230 value is negative, randomization SHALL decrease the *EffectiveDuration* of the *Event*.

3231 For example, if an *Event* is scheduled to conclude at 1:00PM and has a randomizeDuration value of
3232 "-180", a client can select to end the *Event* at 12:59PM, indicating a negative 1 minute randomization. If
3233 the value is positive, randomization SHALL be added to the *Duration* of the *Event*, causing the device to
3234 delay termination of the *Event*.

3235 12.1.4.3 **Application Guidelines / Behavior**

3236 The values required to implement randomization represent relative time in seconds. Therefore, the
3237 signed values can be used to effect either positive or negative (relative to start time / duration)
3238 randomization. Clients that use fixed pseudo random values SHALL scale the applied randomization
3239 based on the range indicated by the given randomizeStart or randomizeDuration. Stated differently,
3240 fixed pseudo random values SHALL indicate a percentage of the randomizeStart or randomizeDuration
3241 to be applied. Fixed pseudo randomization is when a given device has a value that is applied for all
3242 randomizations. Manufacturers SHALL assure that fixed pseudo randomization values are random over
3243 a population of like devices.

3244 Clients that act upon *Events* SHALL support randomization. Clients SHALL apply randomization
3245 attributes when present in an *Event's* attributes. Absence of a randomizeStart or randomizeDuration
3246 value in an *Event* indicates randomization is not to be applied to the commencement or duration of an
3247 *Event*, respectively.

3248 Cancellation of an active *Event* SHALL cause clients to apply the greater of (absolute value of
3249 randomizeStart attribute) and (absolute value of randomizeDuration) to the abbreviated *Event*.

3250 If an *Event* is in progress and user override occurs, the client SHALL respond to the user override
3251 without randomization.

3252 12.1.5 **Multi-Server**

3253 12.1.5.1 **Overview**

3254 This section gives a description of how server and client devices should interact at the application layer
3255 when there are multiple servers of the same function set in the network, particularly in scenarios where
3256 there are multiple servers of a function set for the same commodity. There are several situations where
3257 this may occur; in some situations, one or more of the HAN's servers may be coordinated beyond the
3258 HAN, but this is not required. These guidelines provide methods and guidelines to facilitate orderly and
3259 predictable operations.

3260 12.1.5.2 **Registration**

3261 One key element in a multiple server network is the registration of devices to function set servers.
3262 Devices need the capability of determining which function set servers in the network it should receive
3263 data from and act upon for a particular commodity per function set. Devices MUST complete the
3264 registration and service discovery process for each of the function set servers with which the user
3265 intends it to access information. See the Security, Function Set Assignments and Service Discovery
3266 sections for more details.

3267 There may also be function set servers the device discovers with which it is not registered, or the
3268 function set server does not allow devices to register. In this case, the function set server MAY provide
3269 "public" information (e.g., provided without explicit registration by the user) that devices MAY receive.
3270 Depending on the specific function set guidelines, this may also factor into the application guidelines for
3271 multiple function set server scenarios.

3272 12.1.5.3 **Time**

3273 Time coordination is an important aspect of multiple server scenarios. Each function set server that has a
3274 reference to time SHALL also serve its respective time to the HAN. Clients SHALL choose a primary
3275 time server in the HAN with which to align its internal time per the Time function set guidelines. See
3276 the Time Function Set section (Section 11.1) for guidelines on dealing with multiple time servers in the
3277 network.

3278 12.1.5.4 **Messaging Function Set**

3279 Devices can obtain text messages from many servers, service providers, and / or other devices. It MAY
3280 be very common to have multiple messaging servers in a HAN. For details of managing multiple
3281 messaging servers see the Messaging section 12.5.

3282 12.1.5.5 **Pricing Function Set**

3283 Devices MAY obtain pricing data from many servers, service providers, and / or other devices. It may
3284 be very common to have multiple Pricing servers in a HAN. For details of managing multiple Pricing
3285 servers see the Pricing section 12.4.

3286 12.1.5.6 **DRLC and DER Control Function**

3287 There may be multiple DRLC and / or DER function set instances in a HAN, each operating
3288 independently. With the ability to act on instances from multiple servers that may not be coordinated in
3289 any way, there is the opportunity for conflicting / overlapping *Events* within a function set that the client
3290 will need to handle. Clients SHALL only report acting on one *Event* at a time and SHALL NOT respond
3291 to multiple DRLC or DER servers that they are acting on multiple *Events* for that function set
3292 simultaneously. If clients are getting *Events* from multiple servers of the same function set they SHALL
3293 detect duplicate *Events* by comparing the mRIDs of the *Events*. If a duplicate is detected the client MAY
3294 respond to either, but, if requested, one response is REQUIRED.

3295 When devices are registered to one or more DRLC servers, they SHALL NOT act upon any public
 3296 DRLC servers that are present in the HAN or become available.

3297 When devices are registered to one or more DER Control servers, they SHALL NOT act upon any
 3298 public DER Control servers that are present in the HAN or become available.

3299 Clients SHALL determine the primacy of DRLC and DER Control based on the following in order of
 3300 precedence:

3301 1) Servers SHALL indicate their primacy in the primacy element of the function set instance. See
 3302 schema [ZB 13-0201] definition of PrimacyType for possible values.

3303 2) Clients SHALL prioritize execution of DRLC and DER function set *Events* with different
 3304 PrimacyType attributes using the following guidelines:

3305 0 supersedes 1

3306 1 supersedes 2

3307 2 supersedes 3

3308 If two instances are received with the same priority, then normal Event Rules and
 3309 Guidelines apply (e.g., superseding based on scheduling).

3310 3) If a client is executing an *Event* from one server and decides to execute an *Event* from a
 3311 different server per the application guidelines and if the superseded *Event* responseRequired
 3312 indicates, the client SHALL send a response with "Aborted due to an Alternate Provider Event"
 3313 Response.status for the *Event* that was superseded.

3314 This mechanism and these guidelines allow service providers to coordinate amongst themselves or for
 3315 regulatory / legislative entities to enforce coordination. Reputable service providers should appropriately
 3316 self-select their PrimacyType attribute so as not to disadvantage the consumer's participation in their
 3317 contracted DR or DER Control service providers programs. This method does not prevent non-reputable
 3318 providers from misrepresenting their communications.

3319 12.2 Demand Response and Load Control

3320 12.2.1 Overview

3321 This function set provides an interface for Demand Response and Load Control (DRLC). Client devices
 3322 of this function set include thermostats and devices that support load control. Server devices of this
 3323 function set include ESIs, premises energy management systems that may be acting as a proxy for
 3324 upstream demand response / load control management systems and subsequent data stores.

3325 Servers expose load control events to client devices through resources called "EndDeviceControls"
 3326 (EDC). All EDC instances expose attributes that allow devices to respond to events that are explicitly
 3327 targeted at their device type. For example, an EDC may contain an Offset object indicating a degree
 3328 offset to be applied by a Smart Thermostat. The EDC will also expose necessary attributes that load
 3329 control client devices will need in order to process an event. These include Start Time and Duration, as
 3330 well as an indication on the need for randomization of the start time or duration of the event.

3331 12.2.2 List Ordering

3332 **Table 12-1 Demand Response and Load Control List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
---------------	-------------	---------------	--------------

Resource Name	Primary Key	Secondary Key	Tertiary Key
DemandResponseProgram	Primacy (ascending)	mRID (descending)	NA
EndDeviceControl and ActiveEndDeviceControl	interval.start (ascending)	creationTime (descending)	mRID (descending)

3333 12.2.3 **Application Guidelines / Behavior**3334 12.2.3.1 **DemandResponseProgram**

3335 Multiple programs can be created to target different types of devices or to offer different types of
 3336 incentives. DemandResponseProgram SHALL use mRID as a unique identifier for each instance of a
 3337 program.

3338 DemandResponseProgram server devices SHALL be capable of internally storing and supporting at
 3339 least 1 DemandResponseProgram instance. DemandResponseProgram server devices SHOULD be
 3340 capable of internally sorting and supporting 3 unique DemandResponseProgram instances.

3341 DemandResponseProgram client devices SHALL be capable of internally storing and supporting at least
 3342 1 DemandResponseProgram instance. DemandResponseProgram client devices SHOULD be capable of
 3343 internally storing and supporting 3 unique DemandResponseProgram instances.

3344 12.2.3.2 **EndDeviceControl**

3345 An EndDeviceControl is used to provide control parameters to a DRLC client. An EndDeviceControl
 3346 can always be overridden by the user.

3347 Demand Response / Load Control server devices SHALL be capable of internally storing and supporting
 3348 at least 5 unique EndDeviceControl instances that MAY be distributed between multiple programs.

3349 Demand Response / Load Control server devices SHOULD be capable of internally storing and
 3350 supporting at least of 10 unique EndDeviceControl instances.

3351 Demand Response / Load Control client devices SHALL be capable of internally storing and supporting
 3352 at least 3 unique EndDeviceControl instances. Demand Response / Load Control client devices
 3353 SHOULD be capable of internally storing and supporting at least 5 unique EndDeviceControl instances.
 3354 As a highly RECOMMENDED recovery mechanism, when a maximum of storage of events has been
 3355 reached and additional EndDeviceControl instances are available on the server(s), clients SHOULD
 3356 prioritize local storage and give preference to EndDeviceControls with start times in the near future and
 3357 to events that have been flagged as mandatory.

3358 12.2.3.3 **Rules and Guidelines**

3359 Note that the rules and guidelines detailed below apply to DRLC client devices that change their
 3360 consumption behavior based on the demand response signals received. Display only DRLC client
 3361 devices are not required to comply with the normative statements presented below.

3362 If an EndDeviceControl includes more than one control parameter, the DRLC client is free to utilize
 3363 applicable parameters of its choice. If the DRLC client is capable of supporting multiple parameters
 3364 from the ones that have been included in the event, it SHOULD select the parameter that best fits its
 3365 functionality.

3366 12.2.3.3.1 **availabilityUpdateChangePercentThreshold/ availabilityUpdateChangePowerThreshold**

3367 When Demand Response / Load Control servers support the LoadShedAvailability resource, they
 3368 SHALL use either the availabilityUpdateChangePercentThreshold or
 3369 availabilityUpdateChangePowerThreshold to indicate the threshold for which a client is required to
 3370 update its current load shed ability. The availabilityUpdateChangePercentThreshold is used to indicate a

3371 percent change in load shed availability that SHALL trigger an update from the client. The
3372 availabilityUpdateChangePowerThreshold is used to indicate an absolute change in available load shed
3373 capability that SHALL trigger an update from the client. If no
3374 availabilityUpdateChangePercentThreshold or availabilityUpdateChangePowerThreshold is specified
3375 then clients SHALL NOT POST to their LoadShedAvailability for that program. If the server provides
3376 both, the client SHALL update its current load shed availability when either threshold is crossed.

3377 12.2.3.3.2 **drProgramMandatory**

3378 Events flagged as drProgramMandatory are strongly RECOMMENDED to be acted upon.

3379 If a user attempts to opt-out of an EndDeviceControl with the drProgramMandatory flag set, clients
3380 SHALL require an extra acknowledgment from the user confirming the desire to opt-out. Client devices
3381 MAY allow the user to opt out of EndDeviceControls even if the drProgramMandatory flag is true for
3382 the given event. Clients SHOULD present a warning to indicate that this event has been flagged as
3383 mandatory by their service provider. For example, the client can present a screen stating that "Utility X
3384 has marked this event as mandatory and an opt-out can lead to financial penalties as agreed upon in the
3385 contract with Utility X."

3386 12.2.3.3.3 **overrideDuration**

3387 After the overrideDuration time of an EndDeviceControl has elapsed, the client device SHALL return to
3388 execution of the EndDeviceControl for the remaining Effective Scheduled Period.

3389 Client devices MAY allow users to override an EndDeviceControl for a longer duration than event
3390 overrideDuration, in which case they SHOULD provide a warning for non-compliance if the
3391 drProgramMandatory flag is set to true.

3392 12.2.3.3.4 **DateTimeInterval**

3393 The duration of the EndDeviceControl is provided in seconds using the duration attribute. Events
3394 SHALL NOT be longer than 86,400 seconds (1 day).

3395 12.2.3.3.5 **SetPoint**

3396 When both a Temperature Offset and a Temperature Set Point are provided, the device MAY use either
3397 as defined by the device manufacturer.

3398 In some cases an EDC MAY include an Offset or Set Point that will cause the device to increase its
3399 energy consumption. This may be done as a form of pre-heating or pre-cooling before an event that
3400 reduces consumption is dispatched. This type of EDC SHALL set the loadShiftForward flag to "True".

3401 12.2.3.4 **Response**

3402 When overriding an event, client devices SHOULD provide a duration for the override using the
3403 drOverrideDuration attribute found in the DrResponse object. This is useful for service providers and
3404 EMS in understanding for how long the client device will override the event and when it can expect the
3405 client device to return to shedding load.

3406 Additional guidelines on how the Response resource operates are provided in the Response Function Set
3407 in Section 10.7.

3408 12.2.3.5 **LoadShedAvailability**

3409 When clients support the LoadShedAvailability resource they SHALL POST their ability to shed load
3410 when first connected to a server that supports the resource. If the commitment is related to a program,
3411 the client SHALL provide a link to the specific DemandResponseProgram. Clients SHALL update their
3412 availability based on the update thresholds provided by the program where the load shed is being
3413 committed. If no thresholds are specified by the server, then clients SHALL NOT POST updates to their
3414 LoadShedAvailability for the provider of that program.

3415 Clients reporting their LoadShedAvailability to multiple DemandResponsePrograms SHALL be capable
 3416 of individually providing the committed power or percentage to each program for a reported duration.
 3417 For example, a client capable of shedding 2kW load SHALL NOT report 2kW of availability to two
 3418 separate programs, instead it SHALL divide its availability between the two. Client devices already
 3419 shedding load and reporting on additional availability, SHALL be aware of their current state of
 3420 consumption, as any new event requiring energy reduction SHALL be applied to the device's current
 3421 baseline of consumption.

3422 12.2.4 **LogEvents**

3423 There are no LogEvents generated by this function set.

3424 12.3 **Metering Function Set**

3425 12.3.1 **Overview**

3426 The Metering function set provides interfaces to exchange commodity measurement information such as
 3427 reading types and meter readings between HAN devices. Examples of Meter Flows and XML payloads
 3428 are listed in Appendix C.

3429 Each Metering function set server may have a UsagePointList resource containing resources for local
 3430 meters and metering data mirrored for other devices. One possible scenario is that two electric meters
 3431 exist in a HAN. Both have a UsagePoint resource. Electric Meter #1 (e.g., ESI integrated) has a
 3432 UsagePoint list which contains for example /upt/0 (meter itself) and /upt/1 (mirrored gas meter). Electric
 3433 Meter #2 also has ESI integrated and has for example a /upt list which contains only one instance /upt/0
 3434 (meter itself) but no other meter mirrored to it. Since both UsagePoints are visible, a HAN device that
 3435 does service discovery will find both UsagePoint servers and it then has to decide which UsagePoint
 3436 server to query based on server's information. Note that although there are two /upt/0 instances in this
 3437 case, they are two different servers with different hostnames and / or IP addresses.

3438 12.3.2 **List Ordering**

3439 **Table 12-2 Metering List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
UsagePoint	mRID (descending)	NA	NA
MeterReading	mRID (descending)	NA	NA
ReadingSet	timePeriod.start (descending)	mRID (descending)	NA
Reading	localID (ascending)	consumptionBlock (ascending)	touTier (ascending)

3440 12.3.3 **Application Guidelines / Behavior**

3441 The Metering function set resource hierarchy starts with a list of Usage Points. A Usage Point is an
 3442 abstraction for a point of exchange. This could be represented as a physical meter or a fixed load like a
 3443 street lamp or a virtual metering point as accomplished with transformer or line loss compensation
 3444 algorithms. Each Usage Point then has one or more Meter Readings. Meter Readings serve as an
 3445 aggregation for the ReadingType, a possible Reading and possible Reading Sets. The ReadingType is
 3446 constructed based on subsets and extensions of IEC 61968-9 Annex C – Reading Types [61968]. Only
 3447 relevant IEC fields are listed in the XML schema described in [ZB 13-0201] and SEP 2 UML model. It
 3448 may be beneficial for implementers to obtain a copy of this IEC document in order to better understand

3449 the meanings and uses of the IEC 61968-9 Reading Types [61968]. See the following text for uses of the
 3450 ReadingSets and Readings.

3451 In the following text, the terms "current" and "present" refer to the values at the time the resource is
 3452 read. In terms of ReadingSets the terms refer to the ReadingSet that is being built / filled at the time of
 3453 reading. While a ReadingSet is in this state, ReadingSet.timePeriod.start SHALL be when the
 3454 ReadingSet starts recording its first value and that ReadingSet.timePeriod.duration SHALL grow each
 3455 time the ReadingSet is updated. The ReadingSet.mRID field SHALL be assigned a value of
 3456 0xFFFFFFFFXXXXXXXXXXXXXX[XXXXXXXX] (Where [XXXXXXXX] is replaced by the
 3457 manufacturer's PEN) while the data is being recorded and changed to an appropriate mRID when the
 3458 ReadingSet is complete.

3459 A Metering function set instance that provides instantaneous demand data SHALL serve a Meter
 3460 Reading resource (and subordinate resources) with the following properties:

- 3461 • SHALL contain a ReadingTypeLink element that points to a ReadingType resource that
 3462 matches the InstantaneousDemand definition from Table 12-3.
- 3463 • SHALL contain a ReadingLink element that points to a Reading resource that contains the
 3464 instantaneous demand value.
- 3465 • SHALL NOT contain a ReadingSetListLink element.

3466 A Metering function set instance that provides summation delivered data SHALL serve a Meter Reading
 3467 resource (and subordinate resources) with the following properties:

- 3468 • SHALL contain a ReadingTypeLink element that points to a ReadingType resource that
 3469 matches the SummationDelivered definition from Table 12-3.
 - 3470 ○ The ReadingType resource SHALL specify the number of TOU tiers and / or
 3471 consumption blocks, if any, that the metering instance provides.
- 3472 • SHALL contain a ReadingSetListLink element that points to a ReadingSetList resource.
 - 3473 ○ When metering data is present, the ReadingSetList SHALL contain at least one
 3474 ReadingSet resource, which corresponds to the present summation delivered data.
 - 3475 ■ The ReadingSet resource SHALL contain a ReadingListLink element that
 3476 points to a ReadingList resource. The ReadingList resource SHALL contain
 3477 (number of TOU tiers + 1) multiplied by (number of consumption blocks + 1)
 3478 Reading resources.
 - 3479 ■ The Reading resources in the ReadingList SHALL correspond to the summation
 3480 delivered value for each combination of consumptionBlock=0..(number of
 3481 consumption blocks) and touTier=0..(number of TOU tiers).
 - 3482 ■ The Reading for (consumptionBlock=0, touTier=0) SHALL correspond to the
 3483 total summation delivered.
 - 3484 ■ The Reading for (consumptionBlock=x > 0, touTier=0) SHALL correspond to the
 3485 Block x summation delivered (across all TOU tiers).
 - 3486 ■ The Reading for (consumptionBlock=0, touTier=y > 0) SHALL correspond to the
 3487 TOU Tier y summation delivered (across all consumption blocks).
 - 3488 ■ The Reading for (consumptionBlock=x > 0, touTier=y > 0) SHALL correspond
 3489 to the consumptionBlock x, touTier y summation delivered.
 - 3490 • SHALL contain a ReadingLink to a Reading resource that contains the present summation
 3491 delivered, which is semantically equivalent to the Reading for (consumptionBlock=0,
 3492 touTier=0) for the present ReadingList.

3493 A Metering function set instance that provides summation received data, maximum demand delivered
3494 data, maximum demand received data, and / or other reading type data that utilizes TOU tiers and / or
3495 consumption blocks SHALL serve a Meter Reading resource (and subordinate resources) as per the
3496 rules for Summation Delivered above, but with the ReadingType resource matching the appropriate
3497 definition in Table 12-3, and with the Reading values corresponding to the appropriate data type.

3498 A Metering function set instance that provides interval data SHALL serve a Meter Reading resource
3499 (and subordinate resources) with the following properties:

- 3500 • SHALL contain a ReadingTypeLink element that points to a ReadingType resource that
3501 matches an Interval data definition from Table 12-3.
 - 3502 ○ The The ReadingType resource SHALL specify the intervalLength that is the default for
3503 the intervals contained in the ReadingList resource.
- 3504 • SHALL contain a ReadingSetListLink element that points to a ReadingSetList resource.
 - 3505 ○ When metering data is present, the ReadingSetList SHALL contain at least one
3506 ReadingSet resource, which corresponds to the present Interval Block data (the one
3507 currently being filled).
 - 3508 ■ The ReadingSet resource SHALL contain a ReadingListLink element that
3509 points to a ReadingList resource. The ReadingList resource SHALL contain
3510 Reading resources each of which represents a portion (interval) of the
3511 timePeriod of the ReadingSet.
 - 3512 ■ If the duration in the Time Period of the Reading is not equal to the
3513 intervalLength specified in the ReadingType the timePeriod SHALL be
3514 included in the Reading.
 - 3515 • SHALL NOT contain a ReadingLink resource.

3516 The following table provides a list of common Reading Type Definitions with related fields listed.

3517 **Table 12-3 Reading Types.**

ReadingType Element	accumulationBehaviour	calorificValue	commodity	conversionFactor	dataQualifier	flowDirection	intervalLength	kind	numberOfConsumptionBlocks	numberOfTouTiers	phase	powerOfTenMultiplier	subIntervalLength	supplyLimit	tieredConsumptionBlocks	uom
ReadingType Name																
Instantaneous Demand	12		1			1	E	8				O	E		E	38
Summation Delivered	9		1			1	E	12	O	O		O	E			72
Summation Received	9		1			19	E	12	O	O		O	E			72
Max Demand Delivered	6		1		8	1	O	8	O	O		O		E		38
Max Demand Received	6		1		8	19	O	8	O	O		O		E		38
Intervals Delivered	4		1			1	O	12				O	E		E	72
Intervals Received	4		1			19	O	12				O	E		E	72

Blank cells indicate not used for the given ReadingType Name. May be used for other ReadingTypes.

"E" cells indicate SHALL NOT be used for this class of ReadingType no matter commodity.

"O" cells indicate there MAY be value specified.

Note the values in this table, are provided as examples of possible electricity meter ReadingTypes. A metering instance must set these values as appropriate for its commodities. For example, SummationDelivered may apply to gas or water if the commodity is "NaturalGas" with uom value of 42(m³) or "PotableWater" with uom value of 128(US gl) or 134 (liters). Other commodities SHALL indicate appropriate UOMs. A combination of uom and powerOfTenMultiplier are used to represent units with different magnitudes, for example kWh would be represented as uom of 72 and a powerOfTenMultiplier of 3. As for fractional Wh readings, 0.012 Wh can be expressed as powerOfTenMultiplier = -3 & uom = 72 and value=12.

Relevant UOMs and other reading type fields are listed in the XML schema described in [ZB 13-0201].

3519 12.3.4 **LogEvents**

3520 This subsection includes definitions of all LogEvents that may be raised by the metering function set.
 3521 The LogEvent names and codes are summarized in the table below.

3522 **Table 12-4 Metering LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
UPT_CHECK_METER	0x00	SHOULD be issued when check meter alarm occurs
UPT_TAMPER_DETECT	0x01	SHOULD be issued when a tampering is detected
UPT_POWER_QUALITY	0x02	SHOULD be issued when power quality alarm occurs. It is a generic power quality event code
UPT_LEAK_DETECT	0x03	SHOULD be issued when a leak is detected
UPT_SERVICE_DISCONNECT	0x04	SHOULD be issued when service is disconnected
UPT_SERVICE_LIMITED	0x05	SHOULD be issued when service limited alarm occurs
UPT_LOW_VOLTAGE_L1	0x06	SHOULD be issued when low voltage L1 occurs
UPT_HIGH_VOLTAGE_L1	0x07	SHOULD be issued when high voltage L1 occurs
UPT_LOW_VOLTAGE_L2	0x08	SHOULD be issued when low voltage L2 occurs
UPT_HIGH_VOLTAGE_L2	0x09	SHOULD be issued when high voltage L2 occurs
UPT_LOW_VOLTAGE_L3	0x0A	SHOULD be issued when low voltage L3 occurs
UPT_HIGH_VOLTAGE_L3	0x0B	SHOULD be issued when high voltage L3 occurs
UPT_OVER_CURRENT_L1	0x0C	SHOULD be issued when over current L1 occurs
UPT_OVER_CURRENT_L2	0x0D	SHOULD be issued when over current L2 occurs
UPT_OVER_CURRENT_L3	0x0E	SHOULD be issued when over current L3 occurs

LogEvent Name	LogEvent Code	LogEvent Description
UPT_FREQUENCY_TOO_LOW_L1	0x0F	SHOULD be issued when frequency too low L1
UPT_FREQUENCY_TOO_HIGH_L1	0x10	SHOULD be issued when frequency too high L1
UPT_FREQUENCY_TOO_LOW_L2	0x11	SHOULD be issued when frequency too low L2
UPT_FREQUENCY_TOO_HIGH_L2	0x12	SHOULD be issued when frequency too high L2
UPT_FREQUENCY_TOO_LOW_L3	0x13	SHOULD be issued when frequency too low L3
UPT_FREQUENCY_TOO_HIGH_L3	0x14	SHOULD be issued when frequency too high L3
UPT_GROUND_FAULT	0x15	SHOULD be issued when ground fault occurs
UPT_BURST_DETECT	0x16	SHOULD be issued when burst detect alarm occurs
UPT_PRESSURE_TOO_LOW	0x17	SHOULD be issued when pressure too low
UPT_PRESSURE_TOO_HIGH	0x08	SHOULD be issued when pressure too high
UPT_FLOW_SENSOR_COMMUNICATION_ERROR	0x19	SHOULD be issued when flow sensor communication error occurs
UPT_FLOW_SENSOR_MEASUREMENT_FAULT	0x1A	SHOULD be issued when flow sensor measurement fault occurs
UPT_FLOW_SENSOR_REVERSE_FLOW	0x1B	SHOULD be issued when reverse flow is detected
UPT_FLOW_SENSOR_AIR_DETECT	0x1C	SHOULD be issued when flow sensor air detect alarm occurs
UPT_PIPE_EMPTY	0x1D	SHOULD be issued when pipe empty alarm occurs
UPT_INLET_TEMPERATURE_SENSOR_FAULT	0x1E	SHOULD be issued when inlet temperature sensor fault
UPT_OUTLET_TEMPERATURE_SENSOR_FAULT	0x1F	SHOULD be issued when outlet temperature sensor fault

3523 **12.4 Pricing Function Set**

3524 **12.4.1 Overview**

3525 The pricing function set provides the tariff structures communicated by the server and is designed to
 3526 support a variety of tariff types, including:

- 3527 • Flat-rate pricing
- 3528 • Time-of-Use tiers
- 3529 • Consumption blocks
- 3530 • Hourly day-ahead pricing
- 3531 • Real-time pricing
- 3532 • Combinations of the above

3533 The Pricing Function Set supports application-specific tariffs for devices (e.g., PEV, DER), and special
 3534 event-based prices like critical peak price (Note, as per [ZB 13-0201], CPP is treated as just another TOU
 3535 tier).

3536 The Pricing Function Set is designed to stand on its own but can be paired with the Metering, Billing and
 3537 Prepayment function sets to provide additional benefit to users. The Pricing Function Set is not intended
 3538 to provide all the information necessary to represent a premises's bill.

3539 **12.4.2 List Ordering**

3540 **Table 12-5 Pricing List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
TariffProfile	mRID (descending)	NA	NA
RateComponent	mRID (descending)	NA	NA
TimeTariffInterval <i>and</i> ActiveTimeTariffInterval	interval.start (ascending)	creationTime (descending)	mRID (descending)
ConsumptionTariffInterval	startValue (ascending)	NA	NA

3541 **12.4.3 Application Guidelines / Behavior**

3542 **12.4.3.1 TariffProfile**

3543 Pricing servers SHALL be capable of internally storing and supporting at least one TariffProfile instance.

3544 Pricing clients SHALL be capable of internally storing and supporting at least one TariffProfile instance.

3545 Pricing servers SHOULD be capable of internally storing and supporting at least three TariffProfile
 3546 instances (e.g., multiple commodities, multiple service provider tariff options).

3547 Pricing clients SHOULD be capable of internally storing and supporting at least three TariffProfile
3548 instances (e.g., multiple commodities, multiple service provider tariff options).

3549 12.4.3.2 **Rate Component**

3550 Pricing servers SHALL support at least two RateComponent instances for each TariffProfile.

3551 Pricing clients SHALL support at least one RateComponent instance for each TariffProfile. Pricing
3552 clients supporting the use of this resource SHOULD support at least two instances of RateComponent
3553 (e.g., to convey prices for forward energy [consumed by premises] and reverse energy [supplied by
3554 premises]).

3555 12.4.3.3 **TimeTariffInterval**

3556 Rates that do not contain time-differentiated characteristics SHALL create one TimeTariffInterval
3557 instance with a DateTimeInterval of sufficient duration to cover at least the next 24 hours or use the
3558 maximum time value for duration to minimize data transmission.

3559 Pricing servers SHALL support at least two TimeTariffInterval instances per RateComponent instance
3560 (e.g., the active and subsequent TimeTariffInterval instance for a RateComponent).

3561 Pricing servers SHOULD support at least 48 TimeTariffInterval instances for at least one RateComponent
3562 instance.

3563 Pricing servers SHALL provide at most one active TimeTariffInterval per rate component.
3564 TimeTariffIntervals are to be scheduled for each occurrence of a TOU. A given day would have a flow of
3565 TimeTariffIntervals for the TOU rates for that day. A particular TimeTariffInterval instance specifies the
3566 touTier that is in effect during that event's effective time period.

3567 Pricing clients, upon detecting multiple active TimeTariffIntervals, SHALL ignore all but the
3568 TimeTariffInterval with the highest creation time. If this is insufficient to determine a unique active
3569 TimeTariffInterval (e.g., two active instances exist with the same creation time), clients SHALL operate
3570 as if there is no TimeTariffInterval defined for the given time period.

3571 Pricing clients SHALL be capable of internally storing and supporting at least two TimeTariffInterval
3572 instances per RateComponent instance.

3573 Pricing clients SHOULD be capable of internally storing and supporting at least five TimeTariffInterval
3574 instances per RateComponent instance.

3575 The series of TimeTariffInterval instances on a server may contain gaps or breaks where pricing
3576 information is not defined for some time period. This may occur for various reasons, such as when private
3577 information is cleared from the server (e.g., during move-out) or potentially when superseding
3578 TimeTariffIntervals are created by the service provider (however, service providers should take care to
3579 ensure that gaps are not created, by creating additional TimeTariffInterval instances if necessary). The
3580 below guidelines promote common pricing client behavior and reduce the chances of different
3581 implementations displaying different cost information to a user in the event pricing information is
3582 unavailable during a particular period. This could cause users to question the reliability of the data. Rules
3583 for handling these gaps are as follows:

- 3584 • If a pricing client displays the active or scheduled price or calculated instantaneous cost data to
3585 the user, the client SHALL indicate the presence of an unexpected issue with the price data.
3586 • If a pricing client displays calculated running or averaged cost data to the user, the client MAY
3587 continue to display values by excluding the time periods where no price is defined. However, the

3588 client SHALL also indicate the presence of an unexpected issue with the price data for as long as
 3589 there are price gaps during the time period the client uses to calculate these values.

- 3590 • Pricing clients SHALL NOT default to any hardcoded price attribute (e.g., \$0) or use a price
 3591 attribute from another (past / future) TimeTariffInterval for display or calculation. If a pricing
 3592 client displays human-readable pricing information, then they SHALL display a non-numerical
 3593 indicator (e.g., "XX", dashes, "NA").
- 3594 • If the pricing server later makes TimeTariffInterval instances available to fill pricing gaps,
 3595 pricing clients that display calculated cost information MAY recalculate past cost data based on
 3596 the new information.

3597 Pricing clients that are price responsive SHOULD return to normal operational mode (that is, the default
 3598 behavior of the device without any price responsiveness) during time periods where no
 3599 TimeTariffInterval instance is defined. These clients SHOULD provide some notification to the user that
 3600 the active TimeTariffInterval instance price information is unknown.

3601 The ActiveTimeTariffIntervalList only filters out inactive TimeTariffProfile instances; this SHALL NOT
 3602 filter out ConsumptionTariffInterval instances. That is, if a client GETs a TimeTariffInterval from a
 3603 server's ActiveTimeTariffIntervalList, the ConsumptionTariffIntervalList pointed to by that
 3604 TimeTariffInterval's ConsumptionTariffIntervalListLink SHALL contain all ConsumptionTariffInterval
 3605 instances (equal to the number of consumption blocks defined in the ReadingType), not just the one
 3606 active ConsumptionTariffInterval instance.

3607 12.4.3.4 **Interval**

3608 While strongly RECOMMENDED, Pricing clients are NOT REQUIRED to follow the sign of
 3609 randomization for Pricing function set messages. However, Pricing clients SHALL observe the absolute
 3610 value of the randomizeDuration or randomizeStart value for the randomization range when calculating the
 3611 randomization value. This allows more capable price clients to look ahead at scheduled prices (if
 3612 available) and, using knowledge of the client's operating characteristics, determine if it is in the
 3613 customer's best interest to react to the event earlier or later.

3614 If, while a price-responsive client is acting upon a TimeTariffInterval, that TimeTariffInterval is
 3615 cancelled, the client SHALL observe the randomizeDuration value when ceasing action.

3616 12.4.3.5 **ConsumptionTariffInterval**

3617 Pricing servers SHALL be capable of internally storing and supporting at least one
 3618 ConsumptionTariffInterval element per TimeTariffInterval instance.

3619 Pricing servers SHOULD be capable of internally storing and supporting at least five
 3620 ConsumptionTariffInterval elements per TimeTariffInterval instance.

3621 Pricing clients SHOULD be capable of internally storing and supporting at least five
 3622 ConsumptionTariffInterval elements per TimeTariffInterval instance.

3623 A particular TimeTariffInterval instances MAY NOT include a ConsumptionTariffIntervalListLink,
 3624 meaning that ConsumptionTariffIntervals (and therefore the actual price) is not available for that
 3625 TimeTariffInterval. In such a case, price responsive clients would be unable to act upon price; however,
 3626 they MAY be price responsive to the touTier value (if present) in the TimeTariffInterval.

3627 12.4.3.6 **Sleepy Devices / Polling Clients**

3628 It is RECOMMENDED that sleepy pricing client devices send requests to the pricing server on a periodic
 3629 basis. The RECOMMENDED time period for the periodic poll (for sleepy devices or other clients that do
 3630 not or are unable to make use of subscriptions) is no more than once per hour but at least once per 24-

3631 hour period. This ensures the client a high likelihood of receiving the pricing information needed to
3632 manage its operations in a timely fashion while respecting limited network resources.

3633 It is RECOMMENDED that polling pricing client devices request updated information for pending
3634 TimeTariffInterval instances just prior to those TimeTariffInterval instances becoming active (e.g., 5-10
3635 minutes prior, including any negative randomizeStart). This ensures the TimeTariffInterval instance
3636 previously retrieved is still valid and accurate with the latest instance on the server.

3637 12.4.3.7 Deployments with Multiple Pricing Servers

3638 For the purposes of price responsiveness, clients SHOULD only follow one pricing server in the HAN per
3639 commodity. Pricing clients MAY follow multiple Pricing servers for informational display purposes (e.g.,
3640 to compare different providers) or price seeking behavior. More sophisticated devices (e.g., Premises
3641 Energy Management System) MAY follow multiple Pricing servers and make policy based decisions to
3642 dispatch local resources (e.g., Distributed Energy Resource) and / or provide a single Pricing server for
3643 clients it controls based on user preferences.

3644 Registered pricing devices SHALL determine their primary Pricing server via Function Set Assignments
3645 and follow it for the purposes of price responsiveness. It is incumbent upon the user to choose the Pricing
3646 server with which to register the device. Pricing devices SHALL periodically perform service discovery
3647 to find new pricing servers with which it is registered and begin following them for the purposes of price
3648 responsiveness. Pricing clients SHALL unsubscribe or discontinue following the previous Pricing server
3649 for the purposes of price responsiveness. In addition to periodic discovery of new Pricing servers, it is
3650 RECOMMENDED that devices allow a means of de-registration (or return to defaults) so the device can
3651 be manually de-registered and not require the periodic polling time. If a client is not registered to any
3652 Pricing server, the client SHALL use the Primacy value of any discovered public Pricing servers for the
3653 commodity or commodities of interest.

3654 When devices are registered to a Pricing server, they SHALL not act upon any "public" pricing servers
3655 that are present in the HAN or become available.

3656 12.4.3.8 Relative Pricing between Tiers and Blocks

3657 Pricing servers using multiple TOU tiers SHALL associate higher prices with higher touTier values. That
3658 is, the price of any (TOU Tier N, Consumption Block M) SHALL be less than or equal to the price of
3659 (TOU Tier N+1, Consumption Block M). Note that this is only valid for comparing the same
3660 consumptionBlock between two TOU tiers. Servers MAY be configured such that one or more
3661 consumptionBlock prices from TOU tier N are greater than one or more consumptionBlock prices from
3662 TOU tier N+1.

3663 Similarly, there is no restriction regarding the relative prices between consumptionBlock prices within the
3664 same TOU tier. That is, within one TOU tier, the price of consumptionBlock N MAY be greater than the
3665 price of consumptionBlock N+1.

3666 12.4.3.9 Price Responsiveness

3667 Servers that also support EndDevice instances MAY include price response thresholds for a particular end
3668 device. This provides a standard mechanism for end devices without user interfaces to receive
3669 configuration data concerning customer preferences for price responsiveness. Servers MAY provide a
3670 unique PriceResponseCfg resource for each RateComponent resource that server hosts.

3671 Pricing clients that are registered to a particular Pricing program and are acting upon a particular
3672 RateComponent SHOULD check their PriceResponseCfgList for a PriceResponseCfg resource
3673 corresponding to the RateComponent. If a matching PriceResponseCfg is present, pricing clients

3674 SHOULD consume the associated commodity when the price is less than the consumeThreshold value
 3675 (typically used for scenarios such as negative pricing, pre-heating / cooling, and battery charging), and
 3676 SHOULD reduce consumption to the maximum extent possible when the price is above the
 3677 maxReductionThreshold value.

3678 If an appropriate PriceResponseCfg is not present, or the present price is between the consumeThreshold
 3679 and maxReductionThreshold values, pricing clients MAY base responsiveness on comparing the active
 3680 tier value to the total number of tiers as specified in the ReadingType (as per the price-tier relationship
 3681 defined in Section 12.4.3.8).

3682 PriceResponseCfg servers SHALL NOT specify a consumeThreshold that is greater than or equal to the
 3683 maxReductionThreshold. If a pricing client reads a PriceResponseCfg instance where the
 3684 consumeThreshold is greater than or equal to the maxReductionThreshold, the client SHALL ignore the
 3685 erroneous PriceResponseCfg instance.

3686 12.4.4 LogEvents

3687 **Table 12-6 Pricing LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
TP_NO_TTI	0x00	SHOULD be generated by a Pricing client when there is a gap between two TimeTariffInterval instances or if a TimeTariffInterval instance completes and is not followed by another.

3688 **12.5 Messaging Function Set**

3689 **12.5.1 Overview**

3690 This function set provides an interface for a text messaging service. Client devices of this function set
 3691 include In-Premises Displays and other text messaging display devices. Server devices of this function set
 3692 include ESIs or a back office server, depending on system design. The response function set is used in
 3693 conjunction with the messaging function set to allow client devices to confirm the viewing of messages
 3694 and report advanced responses.

3695 Servers expose messages to client devices in the form of separate messaging URIs. Each messaging URI
 3696 instance will contain information that a client device can use to display the message appropriately. For
 3697 example, start time, duration of event, text to display, etc.

3698 **12.5.2 List Ordering**

3699 For list ordering purposes, the MessagingProgramList, TextMessageList and the ActiveTextMessageList
 3700 SHALL be ordered based on the following criteria:

3701 **Table 12-7 Messaging List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
MessagingProgram	mRID (descending)	NA	NA
TextMessage and ActiveTextMessage	dateTimeInterval.start (ascending)	createdDateTime (descending)	mRID (descending)

3702 12.5.3 **Application Guidelines / Behavior**3703 12.5.3.1 **Messaging Program**

3704 MessagingProgram server and client devices SHALL be capable of internally storing and supporting at
3705 least 1 MessagingProgram instance. A MessagingProgram server and client device SHOULD be capable
3706 of supporting 3 unique MessagingProgram instances.

3707 12.5.3.2 **Text Message**

3708 MessagingProgram server devices SHALL be capable of internally storing and supporting at least 1
3709 MessagingProgram instance. MessagingProgram server devices SHOULD be capable of internally storing
3710 and supporting 3 unique TextMessage instances. Additional information for common application features
3711 that the Messaging function set will perform can be found in the Common Application Functionality
3712 section of this document.

3713 Messaging client devices SHALL be capable of internally storing and supporting at least 1 unique
3714 TextMessage instance, per supported MessagingProgram instance. As a highly RECOMMENDED
3715 recovery mechanism, when a maximum of storage of events has been reached and additional
3716 TextMessage instances are available on the server(s), clients SHOULD prioritize local storage and give
3717 preference to TextMessages with start times in the near future and to events with a higher PriorityType.

3718 Devices can obtain text messages from many sources. It may be very common to have multiple
3719 Messaging functions set servers in a HAN. FunctionSetAssignments and MessagingProgram.primacy are
3720 used to indicate which messaging servers in the HAN the device should prioritize. MessagingPrograms
3721 indicated in FunctionSetAssignments SHALL be of higher priority than those found through discovery.
3722 MessagingProgram.priority SHALL be used as a secondary determinant of a messages priority.

3723 12.5.4 **LogEvents**

3724 There are no LogEvents generated by this function set.

3725 12.6 **Billing Function Set**3726 12.6.1 **Overview**

3727 There are several resources that are used to support billing related functions. The billing function set
3728 provides consumption or costs, estimates of future consumption, and / or historical consumption from a
3729 service provider to an end device. In addition to consumption and costs that would be calculated by the
3730 back end systems and shared with the end devices, billing also provides a mechanism to allow the service
3731 provider to push down targets or challenges to help consumers manage their budgets. A target could be a
3732 percentage or fixed value of reduction – possibly chosen by the consumer - to meet within a defined time
3733 frame.

3734 The TargetReading Resource provides a way for a service provider to create challenges or targets for an
3735 end customer to try to achieve within a certain time frame.

3736 The ProjectionReading Resource provides a way for a service provider to provide future projected
3737 consumption or cost for a particular reading type that may be calculated outside of the HAN.

3738 The HistoricalReading Resource provides a way for a service provider to provide historical consumption
3739 or cost for a particular reading type that is verified and possibly corrected at the service provider backend.

3740 The CustomerAccount resource contains information specific to the account such as the currency. The
3741 Customer Account is associated with a list of Customer Agreements.

3742 The Customer Agreement resource contains information about a particular agreement for service, at a
 3743 particular Usage Point and a particular Tariff Profile rate.

3744 A BillingPeriod relates to a timeframe that a commodity is billed on. There may be multiple
 3745 BillingPeriods that relate to different TariffProfiles.

3746 12.6.2 **List Ordering**

3747 **Table 12-8 Billing List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
CustomerAccount	customerName (ascending)	mRID (descending)	N/A
CustomerAgreement	serviceLocation (ascending)	mRID (descending)	N/A
BillingPeriod	interval.start (descending)	href (ascending)	N/A
HistoricalReading	description (ascending)	mRID (descending)	N/A
TargetReading	description (ascending)	mRID (descending)	N/A
ProjectionReading	description (ascending)	mRID (descending)	N/A
BillingReadingSet	timePeriod.start (descending)	mRID (descending)	N/A
BillingReading	timePeriod.start (ascending)	consumptionBlock (ascending)	touTier (ascending)

3748 12.6.3 **Application Guidelines / Behavior**

3749 12.6.3.1 **CustomerAccount and Customer Agreement Resources**

3750 A CustomerAccount is related to one customer (which may be an organization). Each CustomerAccount
 3751 can have multiple CustomerAgreements associated, possibly representing different UsagePoints or
 3752 commodities. Each CustomerAgreement can link the associated UsagePoint (Metering), Prepayment,
 3753 TariffProfile (Pricing) and / or Billing information together. Servers SHALL support at least one
 3754 CustomerAccount and one CustomerAgreement if the Billing function set is implemented, and SHOULD
 3755 support at least three CustomerAgreements.

3756 12.6.3.2 **BillingPeriod Resource**

3757 For each CustomerAgreement, there may be multiple BillingPeriods. Servers implementing the Billing
 3758 function set SHALL support at least one BillingPeriod per CustomerAgreement, and SHOULD support at
 3759 least three BillingPeriods per CustomerAgreement.

3760 12.6.3.3 **TargetReading Resource**

3761 As a good practice, there should be only one TargetReading for a BillingPeriod. The values of the target
 3762 readings will be an absolute measurement similar to a projected reading. If the service provider specifies
 3763 the targets in a percentage reduction or other method it MUST be converted to an absolute value.

3764 An example would be a customer who used 100kWh in a previous month with a reduction target of 10%.
3765 This target would be specified in the TargetReading as 90kWh and it would be left to the device to
3766 calculate the percentage or kWh reduction to which this equates.

3767 **12.6.3.4 ProjectedReading Resource**

3768 Projected Readings are tools for service providers to help a customer understand what their consumption
3769 or cost might be projected into the future if all things within the home stayed fairly similar.

3770 Examples of projected readings are:

- 3771 • Consumption for the billing period.
3772 • Cost of commodity for the billing period.
3773 • On day X the consumption would be Y which would indicate a customer moving into a higher
3774 block tariff and thus a higher rate.
3775 • The different TOU tiers at the end of the billing period based on current consumption habits.

3776 These are just examples. Other projected readings could be created.

3777 **12.6.3.5 HistoricalReading Resource**

3778 Historical Readings are meant to provide a resource so that a service provider can provide consumption or
3779 cost that has occurred in the past and could be validated and corrected by backoffice systems. Examples
3780 of this would be:

- 3781 • Previous day
3782 • Previous month
3783 • Previous billing period
3784 • Previous year
3785 • TOU A, TOU B, etc.
3786 • Block 1, Block 2, etc.
3787 • Cost of consumption for TOU A for the billing period

3788 This is just a sample of what could be used. Because the information will be provided from the backend
3789 system, it can be of billing quality or near billing quality. End devices that read consumption from
3790 metering end points will be allowed to change their local consumption reading to match information from
3791 the billing resource to allow for closer to actual billed consumption. Servers implementing the Billing
3792 function set SHALL support at least one HistoricalReading and one associated ReadingType,
3793 BillingReadingSet, and BillingReading with at least one Charge, and SHOULD support at least three of
3794 each of these, multiplied by parent containers where appropriate (e.g., three HistoricalReadings with three
3795 BillingReadingSets per HistoricalReading for a total of nine BillingReadingSets, etc.)

3796 12.6.3.6 **Deployments with Multiple Billing Servers**

3797 EndDevices that discover multiple Billing servers SHOULD differentiate between those sources on user
 3798 displays of the information, unless the objects have the same mRID.

3799 12.6.4 **LogEvents**

3800 There are no LogEvents generated by this function set.

3801 12.7 **Prepayment Function Set**

3802 12.7.1 **Overview**

3803 The Prepayment function set defines a mechanism for the conditional delivery of services based upon
 3804 outstanding credit or debt. It provides an interface for appropriately privileged clients to view, update or
 3805 act upon account balance information. Accounting in prepayment systems may be measured on a
 3806 monetary basis (e.g., Dollars or Euros remaining) or on a commodity basis (e.g., kilowatt-hours or BTUs
 3807 remaining). A service-providing device (typically a meter) can use the account balance information from
 3808 a prepayment server in combination with consumption and price data to determine if service should
 3809 continue. In some scenarios ("Local" mode), the service-providing device and the prepayment server are
 3810 the same. Alternatively, the continuation of service may be directed by an external prepayment server,
 3811 either in response to local calculation ("ESI" mode) or to an out-of-band signal from the service provider
 3812 ("Central Wallet" mode). Client devices that provide a user interface to the service provider's customers
 3813 may use the prepayment function set to display information about remaining balances or to request
 3814 additional credit (in some scenarios, through the transmission of payment tokens; however, note that SEP
 3815 2 only provides a wrapper for this token data, the mechanism by which tokens are produced and
 3816 consumed is out of scope for this specification).

3817 12.7.2 **List Ordering**

3818 **Table 12-9 Prepayment List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
Prepayment	mRID (descending)	N/A	N/A
Credit Register	effectiveTime (descending)	mRID (descending)	N/A
SupplyInterruptionOverride	interval.start (ascending)	interval.duration (ascending)	N/A

3819

3820 12.7.3 **Application Guidelines / Behavior**

3821 12.7.3.1 **General**

3822 Servers implementing the Prepayment function set SHALL be capable of internally storing and
 3823 supporting at least one Prepayment instance. Prepayment servers SHALL support one and only one
 3824 AccountBalance resource per Prepayment instance.

3825 A server implementing the Prepayment function set SHALL support one and only one
 3826 PrepayOperationStatus resource per Prepayment instance.

3827 Servers implementing the Prepayment function set in Local or ESI mode SHALL be capable of internally
 3828 storing and supporting at least one CreditRegister instance per Prepayment instance.

3829 Prepayment clients MAY POST to the CreditRegisterList to transmit a new CreditRegister transaction.

3830 A Prepayment client MAY PUT to PrepayOperationStatus to switch between using regular credit and
3831 emergency credit. Typically, this interface is used by service customers to tap a backup credit pool when
3832 normal credit is exhausted and cannot be immediately recharged. Prepayment server implementers MAY
3833 apply additional vendor-specific rules around when this mode of operation may be changed (e.g.,
3834 emergency credit might only be allowed when availableCredit is less than or equal to the
3835 creditExpiryLevel).

3836 **12.7.3.2 Prepayment Server / Usage Point Server Communication**

3837 In most Prepayment configurations, client behavior is minimally affected by server state. That is, the
3838 typical client is an IHD that displays the present account and status data, posts new credit transactions,
3839 requests the use of emergency credit, or displays historical transaction data. However, in the ESI
3840 prepayment mode (and in some configurations of the Central Wallet mode), external meters (or other
3841 service-providing devices), which may be Usage Point servers, are also prepayment clients. These clients
3842 are significantly affected by server state. These devices SHALL monitor the server's Operation Status
3843 resource and should react appropriately to changes of the serviceStatus and serviceChange elements. This
3844 reaction includes connecting or disconnecting service.

3845 **12.7.3.3 Mirroring Behavior**

3846 In most Smart Energy 2.0 function sets, mirroring involves behavior similar to the following pattern:

- 3847 1) Device B issues an OPTIONS method to determine if Server A supports a POST to a given
3848 resource list.
- 3849 2) Device B posts its data to the resource list on Server A. Server A creates a mirrored resource for
3850 Device B.
- 3851 3) Client C reads Device B data from the mirrored resource on Server A.

3852 The Prepayment function set, in some deployments, requires additional mirroring behavior. With
3853 Prepayment, mirror instances MAY need to act as mailboxes for the mirrored server, such as in the
3854 following scenario:

- 3855 1) Device B issues an OPTIONS method to determine if Server A supports a POST to a given
3856 resource list.
- 3857 2) Device B posts its data to the resource list on Server A. Server A creates a mirrored resource for
3858 Device B.
- 3859 3) Client C POSTS data to the mirrored resource on Server A.
- 3860 4) Device B reads the mirrored resource on Server A to obtain Client C data.

3861 One use case for this mode is when a sleepy meter is implementing a Prepayment server in local
3862 prepayment mode. This means that the meter will be expecting CreditRegister transactions to be posted
3863 by prepayment clients. However, sleepy devices are unable to receive transmissions while idle, and a
3864 sleepy server will typically be interacting with other HAN devices through a mirrored instance. In this
3865 scenario, prepayment clients SHALL post the CreditRegister transactions to the mirrored instance. The
3866 server hosting the mirrored instance SHOULD maintain the instances of CreditRegister transactions for at
3867 least 72 hours, so that the sleepy meter MAY download and act upon the credit transactions. Note that for

3868 all intents and purposes (including discovery) that to Prepayment clients on the HAN, the mirrored
 3869 instance is the prepayment server; the other clients are likely unaware of the sleepy device.

3870 12.7.4 LogEvents

3871 **Table 12-10 Prepayment LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
PPY_CREDIT_POST_SUCCESS	0x00	SHOULD be issued when a client successfully posts a CreditRegister to the CreditRegisterList.
PPY_CREDIT_POST_FAIL	0x01	SHOULD be issued when a client posts an invalid resource to the CreditRegisterList.

3872 **12.8 Energy Flow Reservation Function Set**

3873 **12.8.1 Overview**

3874 This function set provides an interface for exchange of energy flow (e.g., charge or discharge) reservation
 3875 events. Client devices of this function set include Plug-in Electric Vehicles, Distributed Energy Storage
 3876 devices, and other managed loads that draw large amounts of power. Server devices of this function set
 3877 include ESIs, EVSEs, and EMSs. FlowReservations allow for the scheduling of high demand periods
 3878 such as during fast-charging transactions, to make them run at different times and avoid high aggregated
 3879 demand. Typically, energy rates have penalties, charges, or customer classes for different demand tiers, so
 3880 it is usually least expensive to keep the maximum demand as low as possible. Distribution utilities may
 3881 support this function set to minimize the maximum demand across the distribution system.

3882 Servers accept FlowReservationRequests from client devices by exposing a FlowReservationRequestList
 3883 for each EndDevice. Clients POST a request to transfer a certain amount of energy during a specific
 3884 interval, at a specific rate. Servers create an associated FlowReservationResponse in the EndDevice's
 3885 FlowReservationResponseList. Servers may create superseding events to modify the interval within the
 3886 requested timeframe and update the status to affect client behavior and distribute load across multiple
 3887 reservations. To do this, the server must have knowledge of multiple clients, but can simply approve all
 3888 requests unchanged if there is no other information.

3889 **12.8.2 List Ordering**

3890 **Table 12-11 FlowReservation List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
FlowReservationRequest	interval.start (ascending)	creationTime (descending)	mRID (descending)
FlowReservationResponse	interval.start (ascending)	creationTime (descending)	mRID (descending)

3891 **12.8.3 Application Guidelines / Behavior**

3892 **12.8.3.1 FlowReservationRequest**

3893 A client generates a FlowReservationRequest in order to trigger a FlowReservationResponse event from
 3894 the server.

3895 FlowReservation server and client devices SHALL be capable of internally storing and supporting at least
 3896 one FlowReservationRequest instance and one FlowReservationResponse instance. A FlowReservation

3897 server and client devices SHOULD be capable of internally storing and supporting at least three unique
 3898 FlowReservationRequest instances.

3899 Clients SHALL NOT modify a FlowReservationRequest except to update the associated RequestStatus.
 3900 Clients SHALL update the associated RequestStatus to Cancelled for any FlowReservationRequest that
 3901 they want a server to subsequently disregard. A server SHALL return a 400 (“Bad Request”) response
 3902 code if it receives a PUT of a FlowReservationRequest that contains changes other than RequestStatus.

3903 If a FlowReservationRequest is removed from the server, clients and servers SHALL NOT assume the
 3904 FlowReservationRequest has been cancelled. Servers MAY remove a request as required (e.g., after the
 3905 request has been fulfilled or space is needed). It is the server’s responsibility to manage
 3906 FlowReservationRequests. Servers SHOULD remove FlowReservationRequests when the associated
 3907 FlowReservationResponse expires.

3908 **12.8.3.2 FlowReservationResponse**

3909

3910 FlowReservation server and client devices SHALL be capable of internally storing and supporting at least
 3911 one FlowReservationResponse instance. FlowReservation server and client devices SHOULD be capable
 3912 of internally storing and supporting at least three unique FlowReservationResponse instances.

3913 A FlowReservationResponse SHALL be created in response to each FlowReservationRequest. If a server
 3914 wants to deny a request, it SHALL create a FlowReservationResponse with duration equal to zero.

3915

3916 **12.8.4 LogEvents**

3917 **Table 12-12 Flow Reservation LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
FR_SCHEDULING_ERROR	0x00	SHOULD be issued if the server encounters an error and is unable to schedule reservations normally.

3918 **12.9 Distributed Energy Resources Function Set**

3919 **12.9.1 Overview**

3920 This function set provides an interface to manage Distributed Energy Resources (DER). There are two
 3921 main types of client devices of this function set: generation and storage. Examples of the first type include
 3922 fuel cells, intelligent solar inverters, and backup generation units. Examples of the second type include
 3923 battery storage systems and electric vehicles (which may not be capable of discharging). Server devices of
 3924 this function set include ESIs and premises energy management systems.

3925 Servers host one or more DERPrograms, which in turn expose DERControl events to DER clients.
 3926 DERControl instances contain attributes that allow DER clients to respond to events that are targeted to
 3927 their device type. A DERControl instance also includes scheduling attributes that allow DER clients to
 3928 store and process future events. These attributes include start time and duration, as well an indication of
 3929 the need for randomization of the start and / or duration of the event.

3930 The SEP 2 DER client model is based on the SunSpec Alliance Inverter Control Model [SunSpec] which
 3931 is derived from IEC 61850-90-7 [61850] and [EPRI]. As these specifications are under development at
 3932 the present time, it is likely that differences will exist between the published standards. The reader is

3933 referred to [EPRI], [61850], and [SunSpec] for a detailed description of the DER features referred to in
3934 this function set.

3935 **12.9.2 Terminology and Conventions**

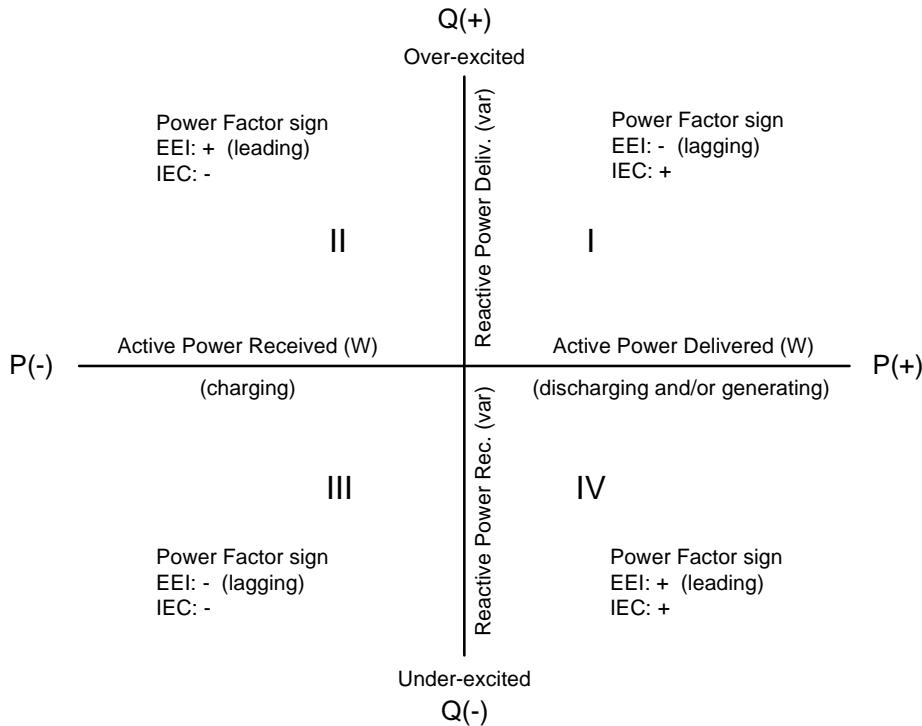
3936 The terminology used to describe the SEP 2 DERControl interface is relative to the DER as a power
3937 producer. A DER described here as a generator *delivers* active AC power for consumption in the
3938 residence or the grid. By convention, a sub-meter connected at the DER accumulates positive energy
3939 usage when the DER is delivering active power. From the utility perspective a DER operating in this
3940 mode may be viewed as a "negative load" and the premises aggregation meter will accumulate energy
3941 usage at a slowed or negative rate.

3942 When the DER has attached storage, it is described here as a load and *receives* active power when in
3943 charging mode. It behaves like a generator and delivers active power when in discharging mode.

3944 By convention, positive active and reactive powers flow in the same direction (here the reference
3945 direction is from the DER to the utility). When a DER is providing positive vars (i.e. behaving like an
3946 over-excited motor or generator) it is said here to be delivering reactive power (VAr).

3947 In addition to the reference frame, the SEP 2 DERControls interface defines the Power Factor (PF) sign
3948 convention used between servers and clients in order to avoid configuration mismatches. DERControl
3949 instances that affect PF assume the EEI [61850] sign convention. Negative ("lagging") PF indicates that
3950 both active and reactive powers are flowing in the same direction (either both delivered or received). Note
3951 that the EEI sign convention treats unity PF as unsigned. It may be necessary for the DER client to locally
3952 translate PF sign before issuing commands to an associated target device.

3953 These relationships are shown in Figure 12-1. At a given point in time, a DER may operate in any one of
3954 the four quadrants depending on its ability to deliver or receive active and reactive power.

**Figure 12-1: Active and Reactive Power Flow Directions as Measured at the DER.****12.9.3 List Ordering****Table 12-13 Distributed Energy Resources List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
DERProgram	primacy (ascending)	mRID (descending)	N/A
DERControl and ActiveDERControl	interval.start (ascending)	creationTime (descending)	mRID (descending)
DERCurve	creationTime (descending)	mRID (descending)	N/A
DER	href (ascending)	N/A	N/A

12.9.4 Application Guidelines / Behavior**12.9.4.1 DERProgram**

Multiple programs can be created to target different types of devices or to offer different types of incentives. A DER client will typically discover its associated DERProgramList through Function Set Assignments.

3965 DERProgram server devices SHALL be capable of internally storing and supporting at least one
 3966 DERProgram instance. DERProgram server devices SHOULD be capable of internally storing and
 3967 supporting three unique DERProgram instances. Each DERProgram instance SHALL be uniquely
 3968 identified by an mRID.

3969 DER client devices SHALL be capable of internally storing and supporting at least one DERProgram
 3970 instance. DER client devices SHOULD be capable of internally storing and supporting two unique
 3971 DERProgram instances.

3972 12.9.4.2 **DERControl**

3973 A DERProgram exposes control parameters to a DER client via *DERControl Events*. At any point in time
 3974 a DER client is managed by a single *DERControl Event*, which SHALL supersede any previous *Event*. A
 3975 DER client MAY reject or partially act upon a *DERControl Event* based on its capabilities. The control
 3976 mode(s) supported by a DER may be determined from its *DERCapability.modesSupported* attribute. If
 3977 there are no active *Events*, the DER client SHALL be managed by the *DefaultDERControl* instance
 3978 exposed by the preferred DERProgram.

3979 DERProgram server devices SHALL be capable of internally storing and supporting at least five unique
 3980 *DERControl* instances, which MAY be distributed among multiple programs. DERProgram server
 3981 devices SHOULD be capable of internally storing and supporting a total of 10 unique *DERControl*
 3982 instances. Each DERProgram SHALL internally store a single *DefaultDERControl* instance that defines
 3983 the default behavior of associated DER clients when no active *Events* are available. Each *DERControl*
 3984 and *DefaultDERControl* instance SHALL be uniquely identified by an mRID.

3985 DER client devices SHALL be capable of internally storing and supporting at least one *DERControl*
 3986 instance and a single *DefaultDERControl* instance for each stored DERProgram. DER client devices
 3987 SHOULD be capable of internally storing and supporting three *DERControl* instances. DER clients
 3988 SHOULD prioritize local storage and give preference to *DERControls* with start times in the near future.

3989 *DERControl* modes are divided into two categories. Immediate control modes (*opModFixedW*,
 3990 *opModFixedPF*, *opModFixedVAr*, *opModFixedFlow*) request a fixed output setting that the DER client
 3991 SHOULD attempt to satisfy. Curve-based control modes allow a DER client to dynamically vary an
 3992 output (dependent variable) as a function of a given input signal (independent variable). For example, an
 3993 intelligent inverter may support the Volt-Watt mode and dynamically control its active power delivery
 3994 based on local voltage measurements. Curve-based control modes are based on *DERCurve* instances.

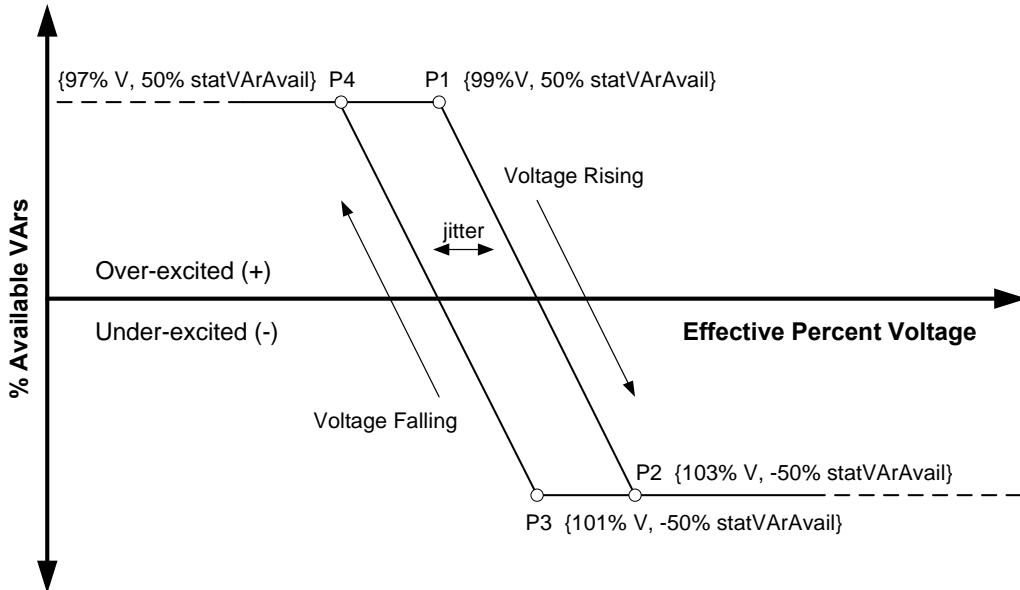
3995 12.9.4.3 **DERCurve**

3996 The *DERCurves* associated with a given DERProgram are grouped under the program's *DERCurveList*
 3997 resource. Each *DERCurve* SHALL contain a defined *curveType* value that associates the curve with a
 3998 given control mode and implicitly defines the units of measure that apply to its *curveData* points.

3999 A *DERCurve* SHALL specify an array of one or more *curveData* points. Watt-PowerFactor curve types
 4000 SHALL specify two dependent variables (*yvalue* plus excitation) per *curveData* point. All other curve
 4001 types define a single independent variable (*xvalue*) and dependent variable (*yvalue*) per *curveData* point.
 4002 See the schema [ZB 13-0201] for details on curve-based DER control modes.

4003 Implementations SHALL support a minimum of four curves having a minimum of 10 points per curve for
 4004 each curve-based *DERControl* mode supported, unless otherwise specified.

4005 As shown in Figure 12-2, an array of points may be used to represent a piecewise linear curve with
 4006 hysteresis. This allows flexibility in defining stable behavior, differences in ramp rates, etc.

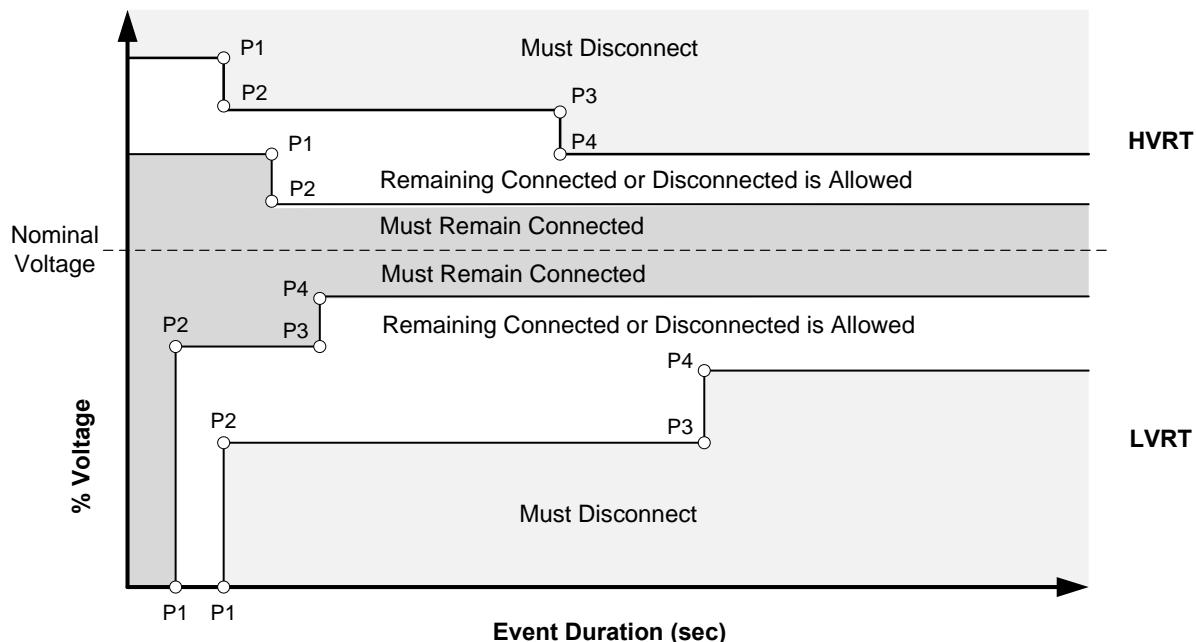
4008
4009

4010

4011

Figure 12-2: Example Volt-VAr Curve and Hysteresis

4012 Figure 12-3 shows the DER operating regions defined by LVRT / HVRT curves. LVRT / HVRT curves
 4013 are assumed to extend horizontally to the left to zero seconds before the first point in the array and to the
 4014 right horizontally after the right-most point in the array.

4015
4016**Figure 12-3: Example Low and High - Voltage Ride Through Curves**

4017 12.9.4.4 **DER Info Resources**

4018 A DER MAY be modeled as one or more DER client instances. For example, a device that consists of a
 4019 solar inverter with attached storage may be modeled as separate or combined generation and storage
 4020 DERs, depending on the complexity of the device's local control interface. The DER resource exposes the
 4021 capability limits of a specific Distributed Energy Resource, as well as basic settings, status, and
 4022 availability. A DER client SHOULD store these resources locally.

4023 The currently executing DERProgram SHALL be referenced by the DER's CurrentDERProgramLink.
 4024 This DERProgram instance SHOULD reference local copies of the active DERControl and DERCurves.

4025 12.9.4.4.1 **DERCapability**

4026 The DER resource exposes the capabilities of a specific Distributed Energy Resource, referred to as its
 4027 "nameplate ratings". Ratings are read-only values established by the DER manufacturer by design or
 4028 manufactured configuration, e.g. the continuous delivered active power rating capability in watts (rtgW),
 4029 and are available by reading the DERCapability resource.

4030 12.9.4.4.2 **DERSettings**

4031 The DERSettings resource provides a means to adjust the operating limits of a DER device as established
 4032 by its nameplate ratings. For example, the active power output (setMaxW) may be reduced or increased as
 4033 a function of attached photo-voltaic panels, condition of the equipment, season of the year, or intended
 4034 use, subject to the maximum limit by rtgMaxW.

4035 The basic settings also include configuration settings related to a specific installation such as setVRef, the
 4036 nominal voltage at the point of common coupling; setVRefOfs, the voltage difference from the point of
 4037 common coupling to the electrical connection point of the inverter; and the set point for the nominal
 4038 frequency. Each rating value in a DER's DERCapability instance MAY have a corresponding setting
 4039 value in its DERSettings instance (which equals the rating value by default). A modified rating SHALL
 4040 have a corresponding setting.

4041 12.9.4.4.3 **DERStatus**

4042 The DER resource references a DERStatus instance that contains basic operational status attributes for the
 4043 DER device. Information such as accumulated generation readings SHOULD be made available by a sub-
 4044 meter referenced by the DER's AssociatedUsagePointLink.

4045 12.9.4.4.4 **DER Availability**

4046 The DERAvailability object is used by client devices to report their availability to deliver reserve active
 4047 and reactive power. It MAY also be exposed instead by devices that are able to report this information on
 4048 behalf of other devices. Duration attributes MAY be provided to indicate how long the generation can be
 4049 sustained.

4050 12.9.5 **DER Client Device Requirements**

4051 DER device architectures are expected to vary widely. Therefore, minimum required functionality is
 4052 based on DER type. A DER instance SHOULD be as simple as possible, but no simpler. For example, if a
 4053 generator type DER can deliver reactive power and supports fixed VAr control mode then it must include
 4054 rtgVAr and setMaxVAr.

4055 If there is a significant difference between delivered and received VAr capability, then the DER MAY
 4056 also include rtgVArNeg and setMaxVArNeg or use rtgVAr to specify a common (minimum) rating for
 4057 both delivered and received VAr. Similarly, if a device includes attached storage and the charging and
 4058 discharging mode rating limits differ significantly then the device can be modeled as separate DERs.

4059 Each unique DER instance SHALL link to a DERCapability instance that SHALL contain type,
 4060 modesSupported, rtgA, and rtgW attributes. The type attribute determines which additional modes and
 4061 attributes are required as shown in the tables below.

4062 The tables are interpreted as follows. If the optionality key in the column Opt1 is "M" (mandatory) then
 4063 the DER SHALL support the control mode(s) listed under modesSupported; if the key is "R"
 4064 (recommended) then the DER SHOULD support the listed control mode; if the key is "O" (optional) then
 4065 the DER MAY support the listed control mode.

4066 If the optionality key in the column Opt2 is "M" (mandatory) then the DER SHALL include the
 4067 attribute(s) listed in the column Related attribute(s); if the key is "R" (recommended) then the DER
 4068 SHOULD support the listed attribute; if the key is "O" (optional) then the DER MAY support the listed
 4069 attribute.

4070 If multiple modes are listed in a given entry, then support for any one of them meets the requirement.
 4071 Each mode will then be individually listed together with its required attributes. If multiple attributes are
 4072 listed in a given entry, then support for any one of them meets the requirement. For example, a storage
 4073 DER that supports only charging (e.g. a PEV) may choose to omit rtgMaxChargeRate and expose its
 4074 maximum charging rate via the rtgW attribute. However, if a storage DER also supports discharge mode
 4075 then it MUST use rtgMaxChargeRate to expose its charging rate and MAY use either
 4076 rtgMaxDischargeRate (if the DER supports simultaneous generation and discharge) or rtgW to expose its
 4077 maximum discharging rate.

4078 **Table 12-14 Modes and Attributes for Generator Type DERs.**

Opt1	modesSupported	Opt2	Related attribute(s)	Notes
M	12 - Fixed W	M	rtgW/setMaxW	Continuous active power output (includes max discharge rate if combined generator / storage type)
		O	rtgVA/setMaxVA	Continuous apparent power out
M	3 - Volt-Watt			
R	13 - Fixed VAr or 14 - Fixed PF			
	13 - Fixed VAr	M	rtgVAr/setMaxVAr	If Fixed VAr mode is supported, then Volt-VAr mode SHOULD be supported
		O	rtgVArNeg/ setMaxVArNeg	SHOULD be included if received and delivered VAr differ significantly
	14 - Fixed PF	M	rtgMinPF/setMinPF	If Fixed PF mode is supported, then Watt-PF mode SHOULD be supported
		O	rtgMinPFNeg/ setMinPFNeg	SHOULD be included if received and delivered VAr differ significantly

4079

4080 Most DERs will typically act as generators, with the possible exception of DERType 81 (electric vehicle).

4081 **Table 12-15 Modes and Attributes for Storage Type DERs.**

Opt1	modesSupported	Opt2	Related attribute(s)	Notes
M	15 - Charge mode	M	rtgMaxChargeRate/ setMaxChargeRate or rtgW/setMaxW	rtgMaxChargeRate is "M" if Discharge mode is supported
		M	rtgWh or rtgAh	Storage capacity
O	16 - Discharge mode	M	rtgMaxChargeRate/ setMaxChargeRate	
		M	rtgMaxDischargeRate/ setMaxDischargeRate or rtgW/setMaxW	rtgMaxDischargeRate is "M" if combined generator / storage type

4082

4083 Storage type DERs include DERType 80 (immobile storage), 81, and 82 (combined PV and storage).
 4084 Note that remote connect and disconnect functions for generator and storage DERs are not control modes
 4085 but settings, and SHOULD be included in DERSettings.

4086 **12.9.6 LogEvents**

4087 There are no LogEvents generated by this function set.

4088 **12.10 Metering Mirror**4089 **12.10.1 Overview**

4090 The Metering Mirror function set provides a mechanism for constrained devices to post metering data to a
 4091 Metering server in a very efficient manner. Great effort has gone into minimizing the number of
 4092 transactions needed to create and maintain the mirroring relationship. Therefore mechanisms and
 4093 structures of this function set differ from other function sets to attain this efficiency.

4094 **12.10.2 List Ordering**4095 **Table 12-16 Metering Mirror List Ordering.**

Resource Name	Primary Key	Secondary Key	Tertiary Key
MirrorUsagePoint	mRID (descending)	NA	NA

4096 **12.10.3 Application Guidelines / Behavior**

4097 A Metering Mirror function set server SHALL NOT advertise support for mirroring unless it has the
 4098 resources available to host at least one additional mirror. The server must have room for at least one
 4099 instance of each of the resources possible under a Usage Point.

4100 The following rules apply to creating and maintaining Metering Mirrors.

- 4101 1) To create a new Metering Mirror the client SHALL POST to the server's MirrorUsagePointList
 4102 (e.g., /mup) for the mirrored usage point.

- 4103 a) This POST SHALL contain at least the information through the definition of
4104 MirrorMeterReadings and ReadingType including the MirrorUsagePoint mRID and
4105 MirrorMeterReading mRIDs.
- 4106 b) The POST MAY also contain MirrorReadingSets and Readings.
- 4107 c) If the mRID of the MirrorUsagePoint is unique (does not match a
4108 MirrorUsagePoint.mRID of an existing MirrorUsagePoint) the response SHALL be
4109 response code 201 (Created), the MirrorUsagePoint URI SHALL be included in the
4110 Location header.
- 4111 d) If the mRID of the MirrorUsagePoint matches an existing MirrorUsagePoint, the new
4112 data SHALL be written over the existing MirrorUsagePoint (and associated UsagePoint)
4113 and the response code SHALL be 204 (No Content), the MirrorUsagePoint URI SHALL
4114 be included in the Location header. If the MirrorUsagePoint contains
4115 MirrorMeterReadings, then the guidance of rules 8 and 9 are to be applied.
- 4116 2) When the Metering Mirror function set server receives a POST it SHALL copy the received data,
4117 including mRIDs, into the normal metering structure to its Metering UsagePoint structure
4118 (e.g., /upt), and it SHALL allocate enough resources to manage the mirror and its data.
- 4119 3) A GET of the resource (MirrorUsagePoint) identified in the response to the initial POST SHALL
4120 return a resource with only the first level elements (i.e., sub-elements and collections are not
4121 included).
- 4122 4) To POST new data to an existing MirrorUsagePoint, the Metering client SHALL POST a
4123 MirrorMeterReading or MirrorMeterReadingList containing MirrorReadingSets and/or Readings
4124 to the resource identified in the Metering server's response to the POST that created the resource
4125 (e.g., /mup/3).
- 4126 5) The Metering Mirror server SHOULD only accept POSTs to a given MirrorUsagePoint from the
4127 client that created the mirror.
- 4128 6) If a POST to the MirrorUsagePoint is of a MirrorMeterReading then a successful response
4129 SHALL contain a Location header indicating the URI of the MeterReading resource under the
4130 associated UsagePoint (e.g., /upt/2/mr/3).
- 4131 7) If a POST to the MirrorUsagePoint is of a MirrorMeterReadingList then a successful response
4132 SHALL contain a Location header indicating the URI of the MeterReadingList under the
4133 associated UsagePoint (e.g., /upt/2/mr).
- 4134 8) In a POST to the MirrorUsagePoint, the mRID attribute of the MirrorMeterReading(s) SHALL be
4135 used by the Metering Mirror server to associate the data in a POST with the MeterReading in the
4136 associated UsagePoint.
- 4137 a) In a POST to the MirrorUsagePoint, if the mRID attribute matches a previous
4138 MirrorMeterReading then the contained MirrorReadingSets SHALL be added to the
4139 associated MeterReading and a contained Reading SHALL replace the existing Reading.
4140 The contents of the MirrorMeterReading shall overwrite the data in the associated
4141 MeterReading.
- 4142 b) In a POST to the MirrorUsagePoint, if the mRID does not match a previous
4143 MirrorMeterReading and it contains a ReadingType, a new MeterReading SHALL be
4144 created under the associated UsagePoint with the new data.

- 4145 c) In a POST to the MirrorUsagePoint, if the mRID does not match a previous
 4146 MirrorMeterReading and there is not a ReadingType then the request SHALL be rejected
 4147 with a response code 400 (Bad Request).
- 4148 9) In a POST to the MirrorUsagePoint, where the request is not rejected, the new data SHALL be
 4149 applied to the related UsagePoint resource structure according to the following:
- 4150 a) If a MirrorReadingSet is received with a duplicate mRID of an existing ReadingSet, and
 4151 it is targeted within the same resource hierarchy, then the new data SHALL replace the
 4152 existing data of the identified ReadingSet.
- 4153 b) If a MirrorReadingSet is received with a unique mRID then the new data SHALL be
 4154 added to the identified ReadingSetList.
- 4155 10) If a client POSTs more data than the Metering Mirror server is willing to accept, the server
 4156 SHALL respond with a response code of 413 (Request Entity Too Large).
- 4157 11) The Metering Mirror server MAY decide when to remove data from the related UsagePoint
 4158 resource structure.

4159 A Metering Mirror function set server MAY implement a timeout mechanism on a mirror. If a Metering
 4160 Mirror function set server does not receive any POSTs from a Metering Mirror function set client for
 4161 more than a specified time the server MAY remove the MirrorUsagePoint resource and its related
 4162 UsagePoint resource. The recommended timeout is 72 hours.

4163 12.10.4 **LogEvents**

4164 **Table 12-17 Metering Mirror LogEvents.**

LogEvent Name	LogEvent Code	LogEvent Description
MUP_MIRROR_EXPIRED	0x00	SHOULD be generated by a server when a Metering Mirror expires.

4165

4166 13 Manufacturer – Specific Proprietary Extensions

4167 13.1 Overview

4168 This section describes rules and mechanisms for interested parties to extend the Smart Energy Profile 2.0
 4169 with proprietary extensions.

4170 It should be noted that as the Smart Energy Profile 2.0 is intended to run over an IP stack, many
 4171 techniques already exist for providing proprietary services over such a stack with various protocols. This
 4172 section is intended for guidance to developers of proprietary extensions that may wish to be similar to the
 4173 design of the Smart Energy Profile 2.0 or add extended elements to the Smart Energy Profile 2.0.

4174 13.2 xmDNS/DNS-SD

4175 Proprietary extensions SHALL NOT be made using the "smartenergy" Service Type.

4176 It is RECOMMENDED that proprietary extensions that wish to use xmDNS/DNS-SD apply for a new
 4177 Service Type with which to operate.

4178 13.3 URIs

4179 As URIs are dynamically discovered and used, proprietary extensions are free to place proprietary
 4180 resources at URIs of their choosing. It is RECOMMENDED that proprietary resources not be placed at
 4181 URIs 'RECOMMENDED' in this specification and in [ZB 13-0201].

4182 13.4 Resources

4183 Proprietary extensions SHALL NOT place any objects, elements, attributes, etc. in the standardized SEP
 4184 XML namespace ("http://zigbee.org/sep") and instead SHALL be placed in a different XML namespace.

4185 The following examples demonstrate allowed and disallowed extensions. In these examples,
 4186 "SEPElement{#}" is used to demonstrate elements that are defined in the SEP schema.
 4187 "MFEElement{#}" and "MFENS" are used to demonstrate elements and namespaces that are proprietary
 4188 extensions respectively.

4189 The example given below demonstrates allowed and disallowed (crossed out) element extensions.

Allowed	Disallowed
<pre><SEPElement1 xmlns="http://zigbee.org/sep" xmlns:MFENS="http://foo.org/mfe"> <SEPElement2>a</SEPElement2> <SEPElement3>b</SEPElement3> <MFENS:MFEElement1>c</MFENS:MFEElement1> </SEPElement1></pre>	<pre><SEPElement1 xmlns="http://zigbee.org/sep"> <SEPElement2>a</SEPElement2> <SEPElement3>b</SEPElement3> <MFEElement1>c</MFEElement1> </SEPElement1></pre>

4190

4191 **Figure 13-1: Allowed and Disallowed Extension**

4192 Proprietary extensions SHALL NOT extend enumerations defined in the SEP 2 schema.

4193 Proprietary extensions made to standardized objects (in a proprietary namespace) defined in the SEP 2
4194 schema SHALL be able to be ignored. That is, a device that does not understand a proprietary extension
4195 can safely ignore the extension.

4196 Proprietary extensions made to standardized objects (in a proprietary namespace) defined in the SEP 2
4197 schema SHALL NOT change the semantics of elements and attributes defined in the SEP 2 schema.

4198 **13.5 DeviceCapabilities Resource**

4199 It is RECOMMENDED that proprietary extensions use a different resource in which to list further
4200 resources.

4201 Should a proprietary extension wish to use the standard DeviceCapabilities resource, it MUST do so
4202 following the same rules as for other resources.

4203 14 Appendix A - Web-Application Description Language (INFORMATIVE)

4204 Note that the WADL (sep_wadl.xml, contained in [ZB 13-0201]) is NORMATIVE. This section presents
4205 a human-friendly view of the information to facilitate understanding.

4206 14.1 Support Resources Section

4207 14.1.1 Device Capability Function Set

4208 14.1.1.1 DeviceCapability Resource

4209 Used to determine the resources available on a server.

4210 Sample URI: /dcap

4211 Request Representation: DeviceCapability

4212 Response Representation: DeviceCapability

4213 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

4214 14.1.2 Self Device Resource Function Set

4215 14.1.2.1 SelfDevice Resource

4216 The device that is providing the services being accessed.

4217 Sample URI: /sdev

4218 Request Representation: SelfDevice

4219 Response Representation: SelfDevice

4220 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

4221 14.1.3 End Device Resource Function Set

4222 14.1.3.1 EndDeviceList Resource

4223 End device resource list.

4224 Sample URI: /eudev

4225 Request Representation: EndDevice

4226 Response Representation: EndDeviceList

4227 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4228 14.1.3.2 EndDevice Resource

4229 End device instance.

4230 Sample URI: /eudev/{id1}

4231 Request Representation: EndDevice

4232 Response Representation: EndDevice

4233 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

4234 14.1.3.3 Registration Resource

4235 Contains registrations related to the indicated device.

4236 Sample URI: /eudev/{id1}/rg

4237 Request Representation: Registration

4238 Response Representation: Registration

4239 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

4240 14.1.3.4 **DeviceStatus Resource**

4241 Contains the current operational state of the associated EndDevice or SelfDevice.

4242 Sample URI: /eudev/{id1}/dstat

4243 Request Representation: DeviceStatus

4244 Response Representation: DeviceStatus

4245 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

4246 14.1.4 **Function Set Assignments Function Set**4247 14.1.4.1 **FunctionSetAssignmentsList Resource**

4248 Contains function set assignments present on the server and/or related to the indicated device.

4249 Sample URI: /eudev/{id1}/fsa

4250 Request Representation: FunctionSetAssignments

4251 Response Representation: FunctionSetAssignmentsList

4252 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4253 14.1.4.2 **FunctionSetAssignments Resource**

4254 Contains links to the specific function set assignments.

4255 Sample URI: /eudev/{id1}/fsa/{id2}

4256 Request Representation: FunctionSetAssignments

4257 Response Representation: FunctionSetAssignments

4258 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4259 14.1.5 **Subscription/Notification Mechanism Function Set**4260 14.1.5.1 **SubscriptionList Resource**

4261 Contains subscriptions related to the indicated device. Documented in Subscription / Notification
4262 Mechanism.

4263 Sample URI: /eudev/{id1}/sub

4264 Request Representation: Subscription

4265 Response Representation: SubscriptionList

4266 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

4267 14.1.5.2 **Subscription Resource**

4268 A specific subscription

4269 Sample URI: /eudev/{id1}/sub/{id2}

4270 Request Representation: Subscription

4271 Response Representation: Subscription

4272 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory

4273 14.1.5.3 **NotificationList Resource**

4274 A list of notifications

4275 Sample URI: /ntfy

4276 Request Representation: Notification

4277 Response Representation: NotificationList

4278 Methods: GET/HEAD: Discouraged, PUT: Error, POST: Mandatory, DELETE: Error

4279 14.1.5.4 **Notification Resource**

4280 A specific notification

4281 Sample URI: /ntfy/{id1}

4282 Request Representation: Notification

4283 Response Representation: Notification

4284 Methods: GET/HEAD: Discouraged, PUT: Error, POST: Error, DELETE: Error

4285 14.1.6 **Response Function Set**4286 14.1.6.1 **ResponseSetList Resource**

4287 List of ResponseSet instances or channels. Devices implementing the ResponseSetList resource MAY support multiple instances of ResponseSet

4289 Sample URI: /rsps

4290 Request Representation: ResponseSet

4291 Response Representation: ResponseSetList

4292 Methods: GET/HEAD: Optional, PUT: Error, POST: Discouraged, DELETE: Error

4293 14.1.6.2 **ResponseSet Resource**

4294 Specific ResponseSet instance. This resource can be thought of as a particular ResponseList or channel.

4295 Sample URI: /rsps/{id1}

4296 Request Representation: ResponseSet

4297 Response Representation: ResponseSet

4298 Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Discouraged

4299 14.1.6.3 **ResponseList Resource**

4300 List of Response instances.

4301 Sample URI: /rsps/{id1}/rsp

4302 Request Representation: Response

4303 Response Representation: ResponseList

4304 Methods: GET/HEAD: Optional, PUT: Error, POST: Mandatory, DELETE: Error

4305 14.1.6.4 **Response Resource**

4306 Specific Response instance.

4307 Sample URI: /rsps/{id1}/rsp/{id2}

4308 Request Representation: Response

4309 Response Representation: Response

4310 Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

4311 14.1.6.5 **PriceResponse Resource**

4312 A specific PriceResponse instance.

4313 Sample URI: /rsps/{id1}/rsp/{id2}

4314 Request Representation: PriceResponse

4315 Response Representation: PriceResponse

4316 Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

4317 14.1.6.6 **TextResponse Resource**

4318 A specific TextMessage Response instance.

4319 Sample URI: /rsps/{id1}/rsp/{id2}

4320 Request Representation: TextResponse

4321 Response Representation: TextResponse

4322 Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

4323 14.1.6.7 **DrResponse Resource**

4324 A specific Demand Response / Load Control EndDeviceControl Response (DrResponse) instance.

4325 Sample URI: /rsps/{id1}/rsp/{id2}

4326 Request Representation: DrResponse

4327 Response Representation: DrResponse

4328 Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

4329 14.2 **Common Resources Section**4330 14.2.1 **Time Function Set**4331 14.2.1.1 **Time Resource**

4332 Provides the Time Resource.

4333 Sample URI: /tm

4334 Request Representation: Time

4335 Response Representation: Time

4336 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

4337 14.2.2 **Device Information Function Set**4338 14.2.2.1 **DeviceInformation Resource**

4339 Device Information of the associated EndDevice or SelfDevice.

4340 Sample URI: /eudev/{id1}/di

4341 Request Representation: DeviceInformation

4342 Response Representation: DeviceInformation

4343 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

4344 14.2.2.2 **SupportedLocaleList Resource**

4345 A List of supported locales for the associated EndDevice or SelfDevice.

4346 Sample URI: /eudev/{id1}/di/loc

4347 Request Representation: SupportedLocale

4348 Response Representation: SupportedLocaleList

4349 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

4350 14.2.2.3 **SupportedLocale Resource**

4351 A specific supported locale for the associated EndDevice or SelfDevice.

4352 Sample URI: /eudev/{id1}/di/loc/{id2}

4353 Request Representation: SupportedLocale

4354 Response Representation: SupportedLocale

4355 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory

4356 14.2.3 **Power Status Function Set**

4357 14.2.3.1 **PowerStatus Resource**

4358 Contains the power status for the associated EndDevice or SelfDevice.

4359 Sample URI: /eudev/{id1}/ps

4360 Request Representation: PowerStatus

4361 Response Representation: PowerStatus

4362 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

4363 14.2.4 **Network Status Function Set**

4364 14.2.4.1 **IPInterfaceList Resource**

4365 List of IPInterface instances on the associated EndDevice or SelfDevice.

4366 Sample URI: /eudev/{id1}/ns

4367 Request Representation: IPInterface

4368 Response Representation: IPInterfaceList

4369 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4370 14.2.4.2 **IPInterface Resource**

4371 Specific IPInterface resource. This resource may be thought of as network status information for a
4372 specific network (IP) layer interface.

4373 Sample URI: /eudev/{id1}/ns/{id2}

4374 Request Representation: IPInterface

4375 Response Representation: IPInterface

4376 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

4377 14.2.4.3 **IPAddrList Resource**

4378 List of IP Addresses

4379 Sample URI: /eudev/{id1}/ns/{id2}/addr

4380 Request Representation: IPAddr

4381 Response Representation: IPAddrList

4382 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4383 14.2.4.4 **IPAddr Resource**

4384 A specific IP Address

4385 Sample URI: /eudev/{id1}/ns/{id2}/addr/{id3}

4386 Request Representation: IPAddr

4387 Response Representation: IPAddr

4388 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

4389 14.2.4.5 **RPLInstanceList Resource**

4390 List of RPL instances that the IPAddr is a member

4391 Sample URI: /eudev/{id1}/ns/{id2}/addr/{id3}/rpl

4392 Request Representation: RPLInstance

4393 Response Representation: RPLInstanceList

- 4394 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error
- 4395 14.2.4.6 **RPLInstance Resource**
- 4396 A specific RPL Instance
- 4397 Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}
- 4398 Request Representation: RPLInstance
- 4399 Response Representation: RPLInstance
- 4400 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional
- 4401 14.2.4.7 **RPLSourceRoutesList Resource**
- 4402 List of RPL source routes
- 4403 Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt
- 4404 Request Representation: RPLSourceRoutes
- 4405 Response Representation: RPLSourceRoutesList
- 4406 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error
- 4407 14.2.4.8 **RPLSourceRoutes Resource**
- 4408 A specific RPL source route
- 4409 Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt/{id5}
- 4410 Request Representation: RPLSourceRoutes
- 4411 Response Representation: RPLSourceRoutes
- 4412 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional
- 4413 14.2.4.9 **LLInterfaceList Resource**
- 4414 List of Link Layer Interfaces
- 4415 Sample URI: /edev/{id1}/ns/{id2}/ll
- 4416 Request Representation: LLInterface
- 4417 Response Representation: LLInterfaceList
- 4418 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error
- 4419 14.2.4.10 **LLInterface Resource**
- 4420 A specific Link Layer Interface
- 4421 Sample URI: /edev/{id1}/ns/{id2}/ll/{id3}
- 4422 Request Representation: LLInterface
- 4423 Response Representation: LLInterface
- 4424 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional
- 4425 14.2.4.11 **NeighborList Resource**
- 4426 List of 802.15.4 neighbors
- 4427 Sample URI: /edev/{id1}/ns/{id2}/ll/{id3}/nbh
- 4428 Request Representation: Neighbor
- 4429 Response Representation: NeighborList
- 4430 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error
- 4431 14.2.4.12 **Neighbor Resource**
- 4432 A specific 802.15.4 neighbor

4433 Sample URI: /edev/{id1}/ns/{id2}/ll/{id3}/nbh/{id4}
4434 Request Representation: Neighbor
4435 Response Representation: Neighbor
4436 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

4437 14.2.5 **Log/Event Log Function Set**

4438 14.2.5.1 **LogEventList Resource**

4439 A List of LogEvent instances.

4440 Sample URI: /edev/{id1}/lel
4441 Request Representation: LogEvent
4442 Response Representation: LogEventList
4443 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

4444 14.2.5.2 **LogEvent Resource**

4445 A specific LogEvent entry from the LogEventList.

4446 Sample URI: /edev/{id1}/lel/{id2}
4447 Request Representation: LogEvent
4448 Response Representation: LogEvent
4449 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Mandatory

4450 14.3 **Smart Energy Resources Section**

4451 14.3.1 **Configuration Resource Function Set**

4452 14.3.1.1 **Configuration Resource**

4453 Contains the configuration settings of the associated EndDevice or SelfDevice.

4454 Sample URI: /edev/{id1}/cfg
4455 Request Representation: Configuration
4456 Response Representation: Configuration
4457 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

4458 14.3.1.2 **PriceResponseCfgList Resource**

4459 Contains a List of price response configuration settings.

4460 Sample URI: /edev/{id1}/prcfg
4461 Request Representation: PriceResponseCfg
4462 Response Representation: PriceResponseCfgList
4463 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

4464 14.3.1.3 **PriceResponseCfg Resource**

4465 Contains the price response configuration settings for this EndDevice associated with a RateComponent.

4466 Sample URI: /edev/{id1}/prcfg/{id2}
4467 Request Representation: PriceResponseCfg
4468 Response Representation: PriceResponseCfg
4469 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory

4470 14.3.2 **Software Download Function Set**

4471 14.3.2.1 **FileList Resource**

4472 A list of files

4473 Sample URI: /file

4474 Request Representation: File

4475 Response Representation: fileList

4476 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4477 14.3.2.2 **File Resource**

4478 A specific file

4479 Sample URI: /file/{id1}

4480 Request Representation: File

4481 Response Representation: File

4482 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

4483 14.3.2.3 **FileStatus Resource**

4484 The file status of a particular file download for the associated EndDevice or SelfDevice.

4485 Sample URI: /eudev/{id1}/fs

4486 Request Representation: FileStatus

4487 Response Representation: FileStatus

4488 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

4489 14.3.3 **Demand Response and Load Control Function Set**

4490 14.3.3.1 **DemandResponseProgramList Resource**

4491 List of DemandResponseProgram instances. Devices implementing the DemandResponseProgramList
4492 resource MAY support multiple instances of DemandResponsePrograms.

4493 Sample URI: /dr

4494 Request Representation: DemandResponseProgram

4495 Response Representation: DemandResponseProgramList

4496 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4497 14.3.3.2 **DemandResponseProgram Resource**

4498 Specific DemandResponseProgram resource. This resource can be thought of as a particular
4499 DemandResponseProgram endpoint.

4500 Sample URI: /dr/{id1}

4501 Request Representation: DemandResponseProgram

4502 Response Representation: DemandResponseProgram

4503 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4504 14.3.3.3 **ActiveEndDeviceControlList Resource**

4505 List of EndDeviceControls that are currently active.

4506 Sample URI: /dr/{id1}/actedc

4507 Request Representation: EndDeviceControl

4508 Response Representation: EndDeviceControlList

4509 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

4510 14.3.3.4 **EndDeviceControlList Resource**

4511 List of EndDeviceControls. Devices implementing the EndDeviceControlList resource MAY support
4512 multiple EndDeviceControls.

4513 Sample URI: /dr/{id1}/edc

4514 Request Representation: EndDeviceControl

4515 Response Representation: EndDeviceControlList

4516 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4517 14.3.3.5 **EndDeviceControl Resource**

4518 Specific EndDeviceControl resource. This resource can be thought of as a particular Demand Response /
4519 Load Control event for a period of time.

4520 Sample URI: /dr/{id1}/edc/{id2}

4521 Request Representation: EndDeviceControl

4522 Response Representation: EndDeviceControl

4523 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4524 14.3.3.6 **LoadShedAvailability Resource**

4525 Allows clients to expose their load shed availability.

4526 Sample URI: /edev/{id1}/lsa

4527 Request Representation: LoadShedAvailability

4528 Response Representation: LoadShedAvailability

4529 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

4530 14.3.4 **Metering Function Set**

4531 14.3.4.1 **UsagePointList Resource**

4532 Usage point resource list. Devices implementing the UsagePointList resource MAY support multiple
4533 instances of usage points.

4534 Sample URI: /upt

4535 Request Representation: UsagePoint

4536 Response Representation: UsagePointList

4537 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4538 14.3.4.2 **UsagePoint Resource**

4539 Usage point instance including links to associated information.

4540 Sample URI: /upt/{id1}

4541 Request Representation: UsagePoint

4542 Response Representation: UsagePoint

4543 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4544 14.3.4.3 **MeterReadingList Resource**

4545 Meter Reading list including explicit URIs for each valid meter reading resource.

4546 Sample URI: /upt/{id1}/mr

4547 Request Representation: MeterReading

4548 Response Representation: MeterReadingList
4549 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4550 14.3.4.4 **MeterReading Resource**

4551 Meter Reading instance which contains ReadingSet and Reading resources

4552 Sample URI: /upt/{id1}/mr/{id2}
4553 Request Representation: MeterReading
4554 Response Representation: MeterReading
4555 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4556 14.3.4.5 **ReadingType Resource**

4557 Meter Reading type

4558 Sample URI: /upt/{id1}/mr/{id2}/rt
4559 Request Representation: ReadingType
4560 Response Representation: ReadingType
4561 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Error

4562 14.3.4.6 **ReadingSetList Resource**

4563 Reading Set list

4564 Sample URI: /upt/{id1}/mr/{id2}/rs
4565 Request Representation: ReadingSet
4566 Response Representation: ReadingSetList
4567 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4568 14.3.4.7 **ReadingSet Resource**

4569 Reading Set instance which contains a list of Reading(s)

4570 Sample URI: /upt/{id1}/mr/{id2}/rs/{id3}
4571 Request Representation: ReadingSet
4572 Response Representation: ReadingSet
4573 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4574 14.3.4.8 **ReadingList Resource**

4575 Reading list of a particular meter reading set.

4576 Sample URI: /upt/{id1}/mr/{id2}/rs/{id3}/r
4577 Request Representation: Reading
4578 Response Representation: ReadingList
4579 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4580 14.3.4.9 **Reading Resource**

4581 Reading instance of a particular meter reading type.

4582 Sample URI: /upt/{id1}/mr/{id2}/rs/{id3}/r/{id4}
4583 Request Representation: Reading
4584 Response Representation: Reading
4585 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

4586 14.3.4.10 **MirrorUsagePointList Resource**

4587 Mirror Usage point (meter) resource list. Devices implementing the MirrorUsagePointList resource may
4588 support multiple instances of meter mirror asset.

4589 Sample URI: /mup

4590 Request Representation: MirrorUsagePoint

4591 Response Representation: MirrorUsagePointList

4592 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

4593 14.3.4.11 **MirrorUsagePoint Resource**

4594 Mirror Usage point instance including resources it supports.

4595 Sample URI: /mup/{id1}

4596 Request Representation: MirrorUsagePoint (PUT), MirrorMeterReading (POST), MirrorMeterReadingList
4597 (POST)

4598 Response Representation: MirrorUsagePoint

4599 Methods: GET/HEAD: Optional, PUT: Mandatory, POST: Mandatory, DELETE: Mandatory

4600 14.3.5 **Pricing Function Set**4601 14.3.5.1 **TariffProfileList Resource**

4602 List of TariffProfile instances.

4603 Sample URI: /tp

4604 Request Representation: TariffProfile

4605 Response Representation: TariffProfileList

4606 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4607 14.3.5.2 **TariffProfile Resource**

4608 Specific TariffProfile instance. Allows clients to obtain information about the rate code.

4609 Sample URI: /tp/{id1}

4610 Request Representation: TariffProfile

4611 Response Representation: TariffProfile

4612 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4613 14.3.5.3 **RateComponentList Resource**

4614 List of RateComponent instances. This list specifies a rate-specific container for the charges associated
4615 with a specific ReadingType, also referenced by the Metering function set.

4616 Sample URI: /tp/{id1}/rc

4617 Request Representation: RateComponent

4618 Response Representation: RateComponentList

4619 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4620 14.3.5.4 **RateComponent Resource**

4621 Specific RateComponent instance. Includes link(s) to the list of TimeTariffIntervals that apply to
4622 referenced ReadingType for the RateComponent instance.

4623 Sample URI: /tp/{id1}/rc/{id2}

4624 Request Representation: RateComponent

4625 Response Representation: RateComponent
4626 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4627 14.3.5.5 **ActiveTimeTariffIntervalList Resource**
4628 The active TimeTariffInterval instance(s) for a particular TariffProfile.
4629 Sample URI: /tp/{id1}/rc/{id2}/acttti
4630 Request Representation: TimeTariffInterval
4631 Response Representation: TimeTariffIntervalList
4632 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

4633 14.3.5.6 **TimeTariffIntervalList Resource**
4634 Collection of TimeTariffInterval instances, including associated ConsumptionTariffInterval.
4635 Sample URI: /tp/{id1}/rc/{id2}/tti
4636 Request Representation: TimeTariffInterval
4637 Response Representation: TimeTariffIntervalList
4638 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4639 14.3.5.7 **TimeTariffInterval Resource**
4640 Specific TimeTariffInterval instance that represents a unique usage interval and associated charges for the
4641 premises.
4642 Sample URI: /tp/{id1}/rc/{id2}/tti/{id3}
4643 Request Representation: TimeTariffInterval
4644 Response Representation: TimeTariffInterval
4645 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4646 14.3.5.8 **ConsumptionTariffIntervalList Resource**
4647 List of ConsumptionTariffInterval instances.
4648 Sample URI: /tp/{id1}/rc/{id2}/tti/{id3}/cti
4649 Request Representation: ConsumptionTariffInterval
4650 Response Representation: ConsumptionTariffIntervalList
4651 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4652 14.3.5.9 **ConsumptionTariffInterval Resource**
4653 Specific ConsumptionTariffInterval instance.
4654 Sample URI: /tp/{id1}/rc/{id2}/tti/{id3}/cti/{id4}
4655 Request Representation: ConsumptionTariffInterval
4656 Response Representation: ConsumptionTariffInterval
4657 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4658 14.3.6 **Messaging Function Set**

4659 14.3.6.1 **MessagingProgramList Resource**
4660 List of MessagingProgram instances or channels. Devices implementing the /msg resource MAY support
4661 multiple instances of MessagingProgram
4662 Sample URI: /msg
4663 Request Representation: MessagingProgram

4664 Response Representation: MessagingProgramList
4665 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4666 14.3.6.2 **MessagingProgram Resource**
4667 Specific MessagingProgram instance. This resource can be thought of as a particular TextMessageList or channel.
4668

4669 Sample URI: /msg/{id1}
4670 Request Representation: MessagingProgram
4671 Response Representation: MessagingProgram
4672 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4673 14.3.6.3 **ActiveTextMessageList Resource**
4674 List of Messages that are currently active.
4675 Sample URI: /msg/{id1}/acttxt
4676 Request Representation: TextMessage
4677 Response Representation: TextMessageList
4678 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

4679 14.3.6.4 **TextMessageList Resource**
4680 A list of TextMessages.
4681 Sample URI: /msg/{id1}/txt
4682 Request Representation: TextMessage
4683 Response Representation: TextMessageList
4684 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4685 14.3.6.5 **TextMessage Resource**
4686 An individual TextMessage.
4687 Sample URI: /msg/{id1}/txt/{id2}
4688 Request Representation: TextMessage
4689 Response Representation: TextMessage
4690 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4691 14.3.7 **Billing Function Set**

4692 14.3.7.1 **CustomerAccountList Resource**
4693 A List of customer accounts
4694 Sample URI: /bill
4695 Request Representation: CustomerAccount
4696 Response Representation: CustomerAccountList
4697 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4698 14.3.7.2 **CustomerAccount Resource**
4699 Customer account information
4700 Sample URI: /bill/{id1}
4701 Request Representation: CustomerAccount
4702 Response Representation: CustomerAccount

- 4703 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged
- 4704 14.3.7.3 **CustomerAgreementList Resource**
- 4705 A list of customer agreements
- 4706 Sample URI: /bill/{id1}/ca
- 4707 Request Representation: CustomerAgreement
- 4708 Response Representation: CustomerAgreementList
- 4709 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error
- 4710 14.3.7.4 **CustomerAgreement Resource**
- 4711 A customer agreement
- 4712 Sample URI: /bill/{id1}/ca/{id2}
- 4713 Request Representation: CustomerAgreement
- 4714 Response Representation: CustomerAgreement
- 4715 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged
- 4716 14.3.7.5 **ActiveBillingPeriodList Resource**
- 4717 A list of active billing periods.
- 4718 Sample URI: /bill/{id1}/ca/{id2}/actbp
- 4719 Request Representation: BillingPeriod
- 4720 Response Representation: BillingPeriodList
- 4721 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error
- 4722 14.3.7.6 **BillingPeriodList Resource**
- 4723 List of BillingPeriods
- 4724 Sample URI: /bill/{id1}/ca/{id2}/bp
- 4725 Request Representation: BillingPeriod
- 4726 Response Representation: BillingPeriodList
- 4727 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error
- 4728 14.3.7.7 **BillingPeriod Resource**
- 4729 Specific Billing Period information
- 4730 Sample URI: /bill/{id1}/ca/{id2}/bp/{id3}
- 4731 Request Representation: BillingPeriod
- 4732 Response Representation: BillingPeriod
- 4733 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged
- 4734 14.3.7.8 **ProjectionReadingList Resource**
- 4735 A list of reading projections.
- 4736 Sample URI: /bill/{id1}/ca/{id2}/pro
- 4737 Request Representation: ProjectionReading
- 4738 Response Representation: ProjectionReadingList
- 4739 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error
- 4740 14.3.7.9 **ProjectionReading Resource**
- 4741 A specific projection reading channel

4742 Sample URI: /bill/{id1}/ca/{id2}/pro/{id3}
4743 Request Representation: ProjectionReading
4744 Response Representation: ProjectionReading
4745 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged
4746 14.3.7.10 **BillingReadingSetList Resource**
4747 A list of billing reading sets
4748 Sample URI: /brs
4749 Request Representation: BillingReadingSet
4750 Response Representation: BillingReadingSetList
4751 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error
4752 14.3.7.11 **BillingReadingSet Resource**
4753 A specific billing reading set
4754 Sample URI: /brs/{id1}
4755 Request Representation: BillingReadingSet
4756 Response Representation: BillingReadingSet
4757 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged
4758 14.3.7.12 **BillingReadingList Resource**
4759 A list of billing readings
4760 Sample URI: /brs/{id1}/br
4761 Request Representation: BillingReading
4762 Response Representation: BillingReadingList
4763 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error
4764 14.3.7.13 **BillingReading Resource**
4765 A specific billing reading
4766 Sample URI: /brs/{id1}/br/{id2}
4767 Request Representation: BillingReading
4768 Response Representation: BillingReading
4769 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged
4770 14.3.7.14 **TargetReadingList Resource**
4771 A list of billing targets.
4772 Sample URI: /bill/{id1}/ca/{id2}/tar
4773 Request Representation: TargetReading
4774 Response Representation: TargetReadingList
4775 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error
4776 14.3.7.15 **TargetReading Resource**
4777 A specific target reading channel
4778 Sample URI: /bill/{id1}/ca/{id2}/tar/{id3}
4779 Request Representation: TargetReading
4780 Response Representation: TargetReading

4781 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4782 14.3.7.16 **HistoricalReadingList Resource**

4783 A list of verified historical readings.

4784 Sample URI: /bill/{id1}/ca/{id2}/ver

4785 Request Representation: HistoricalReading

4786 Response Representation: HistoricalReadingList

4787 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4788 14.3.7.17 **HistoricalReading Resource**

4789 A specific historical reading channel

4790 Sample URI: /bill/{id1}/ca/{id2}/ver/{id3}

4791 Request Representation: HistoricalReading

4792 Response Representation: HistoricalReading

4793 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4794 14.3.7.18 **ServiceSupplier Resource**

4795 A specific service supplier

4796 Sample URI: /bill/{id1}/ss

4797 Request Representation: ServiceSupplier

4798 Response Representation: ServiceSupplier

4799 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4800 14.3.8 **Prepayment Function Set**

4801 14.3.8.1 **PrepaymentList Resource**

4802 A List of Prepayment instances.

4803 Sample URI: /ppy

4804 Request Representation: Prepayment

4805 Response Representation: PrepaymentList

4806 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4807 14.3.8.2 **Prepayment Resource**

4808 A particular Prepayment instance. Provides links to the Account Balance, Credit Register, and Operation Status resources for a particular service.

4810 Sample URI: /ppy/{id1}

4811 Request Representation: Prepayment

4812 Response Representation: Prepayment

4813 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4814 14.3.8.3 **AccountBalance Resource**

4815 Account Balance instance.

4816 Sample URI: /ppy/{id1}/ab

4817 Request Representation: AccountBalance

4818 Response Representation: AccountBalance

4819 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

4820 14.3.8.4 **PrepayOperationStatus Resource**

4821 The Operation Status for the given service. Identifies whether service should be continued. MAY also
4822 include other status information, such as low credit warning.

4823 Sample URI: /ppy/{id1}/os

4824 Request Representation: PrepayOperationStatus

4825 Response Representation: PrepayOperationStatus

4826 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

4827 14.3.8.5 **ActiveSupplyInterruptionOverrideList Resource**

4828 A list of active supply interruption overrides

4829 Sample URI: /ppy/{id1}/actsi

4830 Request Representation: SupplyInterruptionOverride

4831 Response Representation: SupplyInterruptionOverrideList

4832 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

4833 14.3.8.6 **SupplyInterruptionOverrideList Resource**

4834 A List of Supply Interruption Override instances.

4835 Sample URI: /ppy/{id1}/si

4836 Request Representation: SupplyInterruptionOverride

4837 Response Representation: SupplyInterruptionOverrideList

4838 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4839 14.3.8.7 **SupplyInterruptionOverride Resource**

4840 A particular Supply Interruption Override instance. This defines a period of time during which supply
4841 would not be interrupted even if available credit has been exhausted.

4842 Sample URI: /ppy/{id1}/si/{id2}

4843 Request Representation: SupplyInterruptionOverride

4844 Response Representation: SupplyInterruptionOverride

4845 Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

4846 14.3.8.8 **CreditRegisterList Resource**

4847 A List of Credit Register instances. Interface for new credit transactions.

4848 Sample URI: /ppy/{id1}/cr

4849 Request Representation: CreditRegister

4850 Response Representation: CreditRegisterList

4851 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4852 14.3.8.9 **CreditRegister Resource**

4853 A particular Credit Register instance. Records a payment transaction or other credit addition.

4854 Sample URI: /ppy/{id1}/cr/{id2}

4855 Request Representation: CreditRegister

4856 Response Representation: CreditRegister

4857 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4858 14.3.9 **Flow Reservation Function Set**

4859 14.3.9.1 **FlowReservationRequestList Resource**

4860 List of FlowReservationRequests. Devices implementing the FlowReservationRequestList resource
4861 MAY support multiple FlowReservationRequests.

4862 Sample URI: /edev/{id1}/frq

4863 Request Representation: FlowReservationRequest

4864 Response Representation: FlowReservationRequestList

4865 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

4866 14.3.9.2 **FlowReservationRequest Resource**

4867 Specific FlowReservationRequest resource. This resource can be thought of as a particular reservation
4868 event request for fast charging or discharging over a period of time.

4869 Sample URI: /edev/{id1}/frq/{id2}

4870 Request Representation: FlowReservationRequest

4871 Response Representation: FlowReservationRequest

4872 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

4873 14.3.9.3 **FlowReservationResponseList Resource**

4874 List of FlowReservationResponses. Devices implementing the FlowReservationResponseList resource
4875 MAY support multiple FlowReservationResponses.

4876 Sample URI: /edev/{id1}/frp

4877 Request Representation: FlowReservationResponse

4878 Response Representation: FlowReservationResponseList

4879 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

4880 14.3.9.4 **FlowReservationResponse Resource**

4881 Specific FlowReservationResponse resource. This resource can be thought of as a particular reservation
4882 event response for fast charging or discharging over a period of time.

4883 Sample URI: /edev/{id1}/frp/{id2}

4884 Request Representation: FlowReservationResponse

4885 Response Representation: FlowReservationResponse

4886 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

4887 14.3.10 **Distributed Energy Resources Function Set**

4888 14.3.10.1 **DERList Resource**

4889 A list of Distributed Energy Resources

4890 Sample URI: /edev/{id1}/der

4891 Request Representation: DER

4892 Response Representation: DERList

4893 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4894 14.3.10.2 **DER Resource**

4895 The information about a specific Distributed Energy Resource.

4896 Sample URI: /edev/{id1}/der/{id2}

4897 Request Representation: DER
4898 Response Representation: DER
4899 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional
4900 14.3.10.3 **AssociatedUsagePoint Resource**
4901 The usage point associated with this DER instance.
4902 Sample URI: /edev/{id1}/der/{id2}/upt
4903 Request Representation: UsagePoint
4904 Response Representation: UsagePoint
4905 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional
4906 14.3.10.4 **AssociatedDERProgramList Resource**
4907 The List of DERProgram instances associated with this DER.
4908 Sample URI: /edev/{id1}/der/{id2}/derp
4909 Request Representation: DERProgram
4910 Response Representation: DERProgramList
4911 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error
4912 14.3.10.5 **CurrentDERProgram Resource**
4913 The specific DER Control Program being followed by the DER.
4914 Sample URI: /edev/{id1}/der/{id2}/cdp
4915 Request Representation: DERProgram
4916 Response Representation: DERProgram
4917 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional
4918 14.3.10.6 **DERSettings Resource**
4919 The DER settings of the associated EndDevice or SelfDevice.
4920 Sample URI: /edev/{id1}/der/{id2}/derg
4921 Request Representation: DERSettings
4922 Response Representation: DERSettings
4923 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error
4924 14.3.10.7 **DERStatus Resource**
4925 The DER status of the associated EndDevice or SelfDevice.
4926 Sample URI: /edev/{id1}/der/{id2}/ders
4927 Request Representation: DERStatus
4928 Response Representation: DERStatus
4929 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error
4930 14.3.10.8 **DERAvailability Resource**
4931 The DER availability of the associated EndDevice or SelfDevice.
4932 Sample URI: /edev/{id1}/der/{id2}/dera
4933 Request Representation: DERAvailability
4934 Response Representation: DERAvailability
4935 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

4936 14.3.10.9 **DERCapability Resource**

4937 Capabilities of the DER

4938 Sample URI: /eudev/{id1}/der/{id2}/dercap

4939 Request Representation: DERCapability

4940 Response Representation: DERCapability

4941 Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

4942 14.3.10.10 **DERProgramList Resource**

4943 List of DERProgram instances. Devices implementing the DERProgramList resource MAY support
4944 multiple instances of DERPrograms.

4945 Sample URI: /derp

4946 Request Representation: DERProgram

4947 Response Representation: DERProgramList

4948 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4949 14.3.10.11 **DERProgram Resource**

4950 Specific DER Control Program collection resource. This resource can be thought of as a particular
4951 DERProgram endpoint. This representation contains simple management attributes, as well as each
4952 associated resource.

4953 Sample URI: /derp/{id1}

4954 Request Representation: DERProgram

4955 Response Representation: DERProgram

4956 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4957 14.3.10.12 **ActiveDERControlList Resource**

4958 List of DERControls that are currently active.

4959 Sample URI: /derp/{id1}/actderc

4960 Request Representation: DERControl

4961 Response Representation: DERControlList

4962 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

4963 14.3.10.13 **DERControlList Resource**

4964 List of DERControls. Devices implementing the DERControlList resource MAY support multiple
4965 DERControls.

4966 Sample URI: /derp/{id1}/derc

4967 Request Representation: DERControl

4968 Response Representation: DERControlList

4969 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4970 14.3.10.14 **DERControl Resource**

4971 Specific DERControl resource. This resource can be thought of as a particular DER control event for a
4972 period of time.

4973 Sample URI: /derp/{id1}/derc/{id2}

4974 Request Representation: DERControl

4975 Response Representation: DERControl

4976 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4977 14.3.10.15 **DefaultDERControl Resource**

4978 The DefaultDERControl resource. This resource can be thought of as the default DERControl to be used
4979 if no active DERControl event is found.

4980 Sample URI: /derp/{id1}/dderc

4981 Request Representation: DefaultDERControl

4982 Response Representation: DefaultDERControl

4983 Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Error

4984 14.3.10.16 **DERCurveList Resource**

4985 A List of DER curves

4986 Sample URI: /derp/{id1}/dc

4987 Request Representation: DERCurve

4988 Response Representation: DERCurveList

4989 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

4990 14.3.10.17 **DERCurve Resource**

4991 A DER curve instance

4992 Sample URI: /derp/{id1}/dc/{id2}

4993 Request Representation: DERCurve

4994 Response Representation: DERCurve

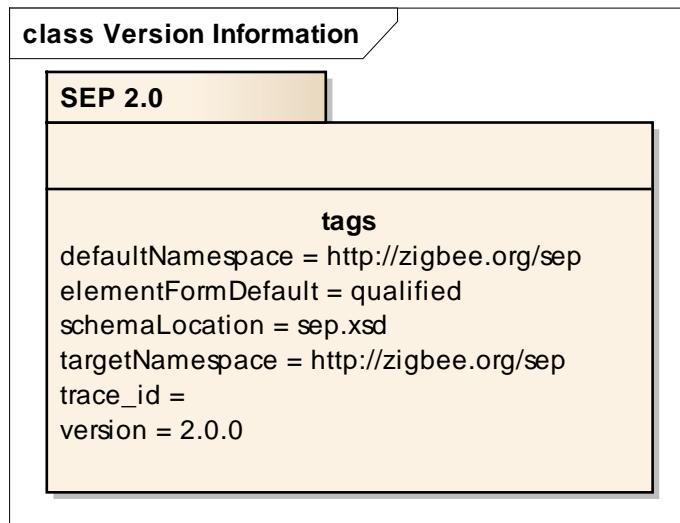
4995 Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

4996 15 Appendix B – SEP 2 Model (INFORMATIVE)

4997 Note that the XML version of the model, contained in the XML Schema definition "sep.xsd" contained in
 4998 [ZB 13-0201] is normative. This section presents a human-friendly view of the information to facilitate
 4999 comments on the content.

5000 15.1 SEP 2 Package

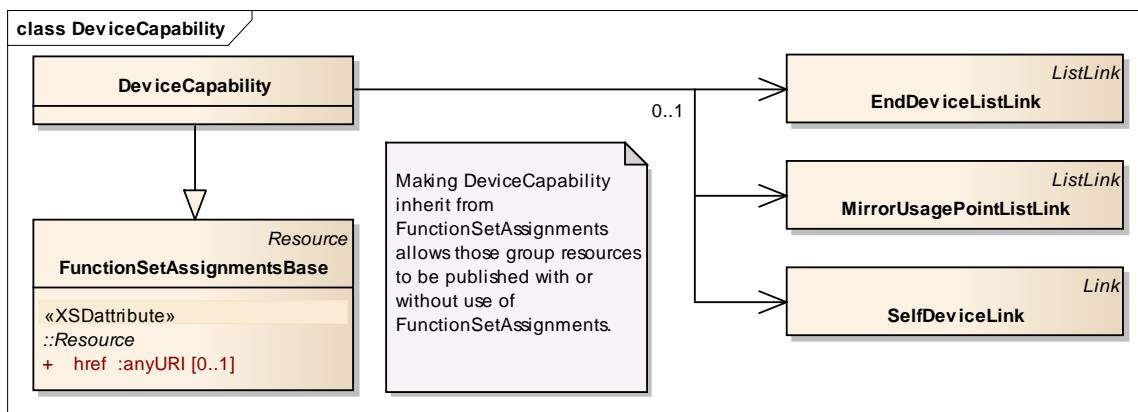
5001 The Smart Energy Profile 2.0 model is organized into function sets, represented by sub-packages.
 5002 However, all structures are defined inside a single namespace.



5003
 5004 **Figure 15-1: Version Information**

5005 15.1.1 DeviceCapability Package

5006 Contains definition of the objects used to convey the resources that are implemented by the publishing
 5007 host.



5008
 5009 **Figure 15-2: DeviceCapability**

5010 **DeviceCapability Object** (FunctionSetAssignmentsBase)

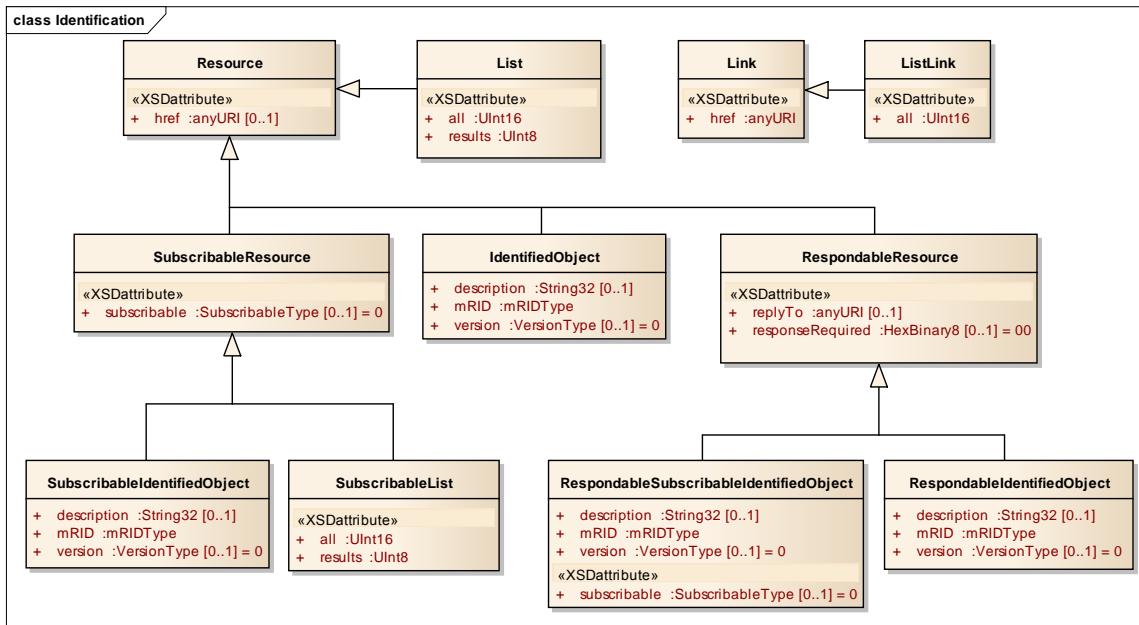
5011 Returned by the URI provided by DNS-SD, to allow clients to find the URIs to the resources in
5012 which they are interested.

5013 **15.1.2 Common Package**

5014 This package contains objects that are used in multiple function sets.

5015 **15.1.2.1 Identification Package**

5016 Contains super-classes that define the attributes common to categories of objects.



5017

5018 **Figure 15-3: Identification**

5019 **IdentifiedObject Object** (Resource)

5020 This is a root class to provide common naming attributes for all classes needing naming attributes

5021 **description attribute** (String32) [0..1]

5022 The description is a human readable text describing or naming the object.

5023 **mRID attribute** (mRIDType)

5024 The global identifier of the object.

5025 **version attribute** (VersionType) [0..1]

5026 Contains the version number of the object. See the type definition for details.

5027 **Link Object** ()

5028 Links provide a reference, via URI, to another resource.

5029 **href attribute** (anyURI) «XSAttribute»

5030 A URI reference.

5031 **List Object** (Resource)

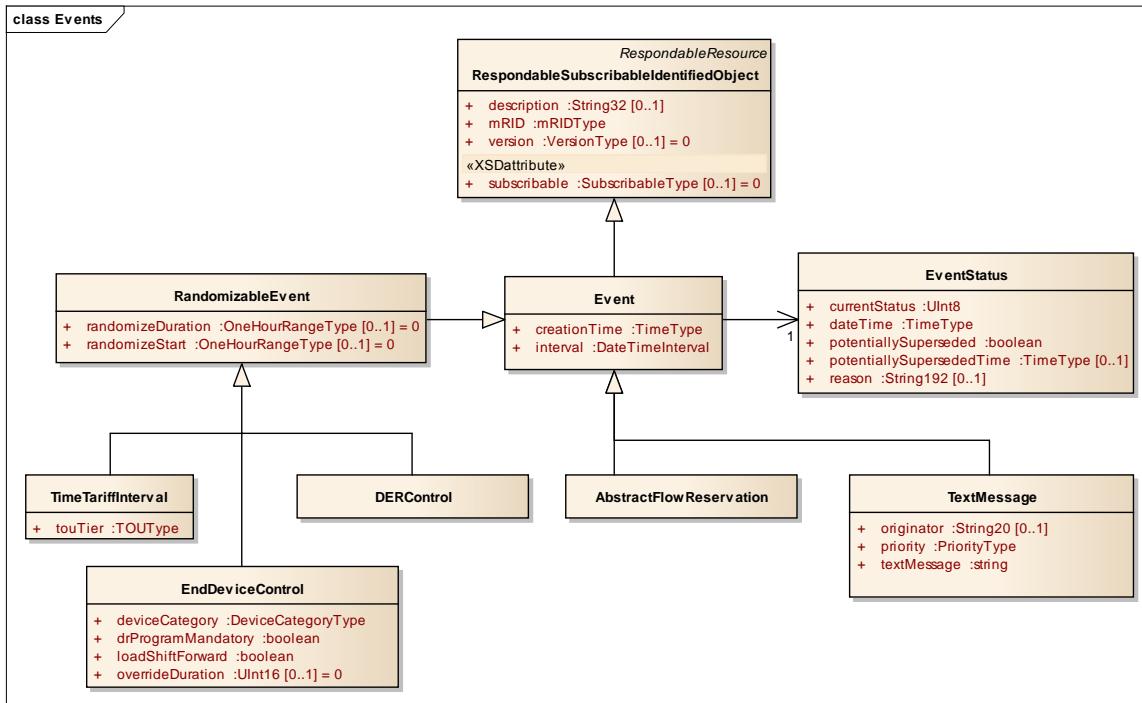
5032 Container to hold a collection of object instances or references. See [ZB 11-0167] Design
5033 Patterns section for additional details.

- 5034 ***all attribute*** (*UInt16*) «*XSDattribute*»
 5035 The number specifying "all" of the items in the list. Required on a response to a GET, ignored otherwise.
- 5036 ***results attribute*** (*UInt8*) «*XSDattribute*»
 5037 Indicates the number of items in this page of results.
- 5038 **ListLink Object** (Link)
 5039 ListLinks provide a reference, via URI, to a List.
- 5040 ***all attribute*** (*UInt16*) «*XSDattribute*»
 5041 Indicates the total number of items in the referenced list.
- 5042 **Resource Object** ()
 5043 A resource is an addressable unit of information, either a collection (List) or instance of an object
 5044 (identifiedObject, or simply, Resource)
- 5045 ***href attribute*** (*anyURI*) [0..1] «*XSDattribute*»
 5046 A reference to the resource address (URI). Required in a response to a GET, ignored otherwise.
- 5047 **RespondableIdentifiedObject Object** (RespondableResource)
 5048 An IdentifiedObject to which a Response can be requested.
- 5049 ***description attribute*** (*String32*) [0..1]
 5050 The description is a human readable text describing or naming the object.
- 5051 ***mRID attribute*** (*mRIDType*)
 5052 The global identifier of the object.
- 5053 ***version attribute*** (*VersionType*) [0..1]
 5054 Contains the version number of the object. See the type definition for details.
- 5055 **RespondableResource Object** (Resource)
 5056 A Resource to which a Response can be requested.
- 5057 ***replyTo attribute*** (*anyURI*) [0..1] «*XSDattribute*»
 5058 A reference to the response resource address (URI). Required on a response to a GET if responseRequired
 5059 is "true".
- 5060 ***responseRequired attribute*** (*HexBinary8*) [0..1] «*XSDattribute*»
 5061 Indicates whether or not a response is required upon receipt, creation or update of this resource.
 5062 Responses shall be posted to the collection specified in "replyTo".
- 5063 If the resource has a deviceCategory field, devices that match one or more of the device types indicated in
 5064 deviceCategory SHALL respond according to the rules listed below. If the category does not match, the
 5065 device SHALL NOT respond. If the resource does not have a deviceCategory field, a device receiving the
 5066 resource SHALL respond according to the rules listed below.
- 5067 Value encoded as hex according to the following bit assignments, any combination is possible.
- 5068 See Table 10-10 for the list of appropriate Response status codes to be sent for these purposes.
- 5069 0 - End device shall indicate that message was received
- 5070 1 - End device shall indicate specific response.
- 5071 2 - End user / customer response is required.

- 5072 All other values reserved.
- 5073 **RespondableSubscribableIdentifiedObject Object** (RespondableResource)
5074 An IdentifiedObject to which a Response can be requested.
- description attribute** (String32) [0..1]
5075 The description is a human readable text describing or naming the object.
- mRID attribute** (mRIDType)
5077 The global identifier of the object.
- subscribable attribute** (SubscribableType) [0..1] «XSDattribute»
5079 Indicates whether or not subscriptions are supported for this resource, and whether or not conditional
5080 (thresholds) are supported. If not specified, is "not subscribable" (0).
- version attribute** (VersionType) [0..1]
5082 Contains the version number of the object. See the type definition for details.
- 5084 **SubscribableIdentifiedObject Object** (SubscribableResource)
5085 An IdentifiedObject to which a Subscription can be requested.
- description attribute** (String32) [0..1]
5086 The description is a human readable text describing or naming the object.
- mRID attribute** (mRIDType)
5088 The global identifier of the object.
- version attribute** (VersionType) [0..1]
5090 Contains the version number of the object. See the type definition for details.
- 5092 **SubscribableList Object** (SubscribableResource)
5093 A List to which a Subscription can be requested.
- all attribute** (UInt16) «XSDattribute»
5094 The number specifying "all" of the items in the list. Required on GET, ignored otherwise.
- results attribute** (UInt8) «XSDattribute»
5096 Indicates the number of items in this page of results.
- 5098 **SubscribableResource Object** (Resource)
5099 A Resource to which a Subscription can be requested.
- subscribable attribute** (SubscribableType) [0..1] «XSDattribute»
5100 Indicates whether or not subscriptions are supported for this resource, and whether or not conditional
5101 (thresholds) are supported. If not specified, is "not subscribable" (0).

5103 15.1.2.2 **Objects Package**

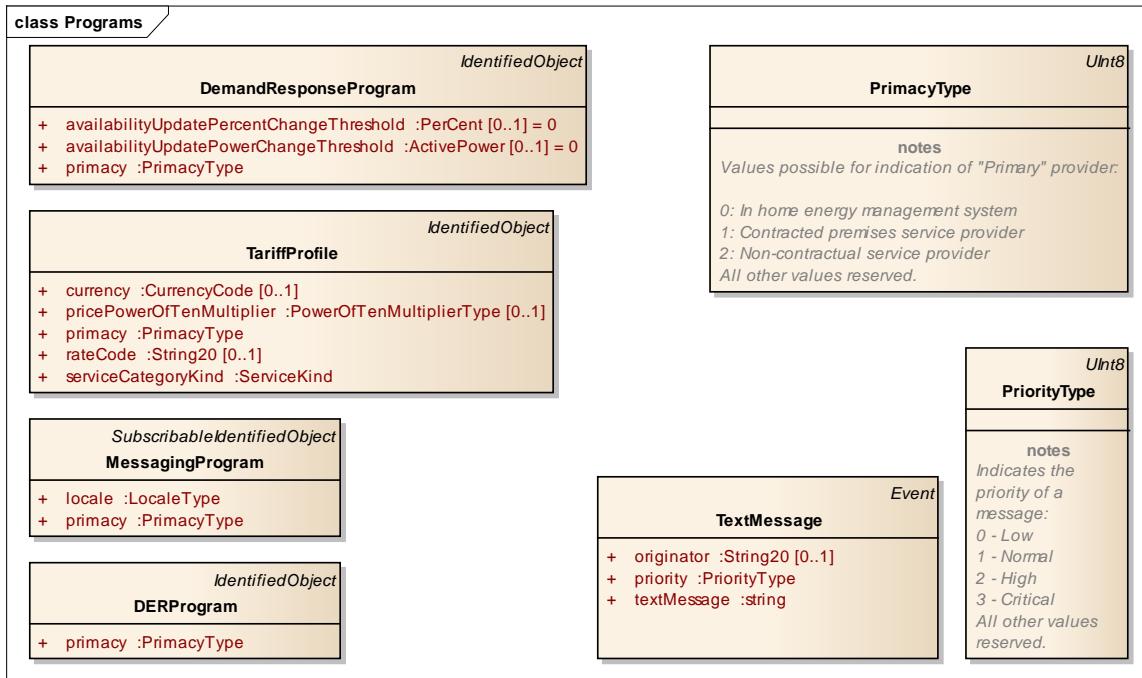
5104 Contains definitions of objects used by multiple function sets.



5105

5106

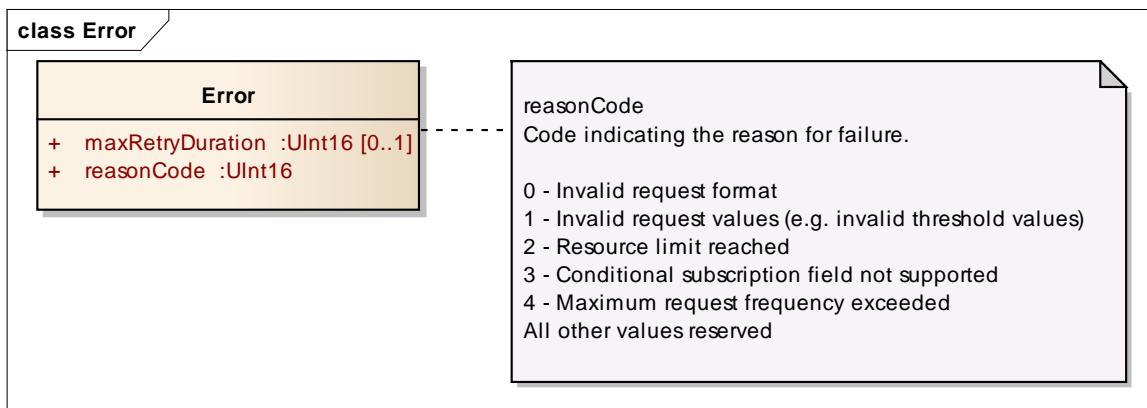
Figure 15-4: Events



5107

5108

Figure 15-5: Programs



5109

Figure 15-6: Error

5111

Error Object ()

5112 Contains information about the nature of an error if a request could not be completed
 5113 successfully.

5114 ***maxRetryDuration attribute (UInt16) [0..1]***

5115 Contains the number of seconds the client SHOULD wait before retrying the request.

5116 ***reasonCode attribute (UInt16)***

5117 Code indicating the reason for failure.

5118 0 - Invalid request format

5119 1 - Invalid request values (e.g. invalid threshold values)

5120 2 - Resource limit reached

5121 3 - Conditional subscription field not supported

5122 4 - Maximum request frequency exceeded

5123 All other values reserved

Event Object (RespondableSubscribableIdentifiedObject)

5124 An Event indicates information that applies to a particular period of time. Events SHALL be
 5125 executed relative to the time of the server, as described in the Time function set section 11.1.

5126 ***creationTime attribute (TimeType)***

5127 The time at which the Event was created.

5128 ***interval attribute (DateTimeInterval)***

5129 The period during which the Event applies.

EventStatus Object ()

5130 Current status information relevant to a specific object. The Status object is used to indicate the
 5131 current status of an Event. Devices can read the containing resource (e.g. TextMessage) to get the
 5132 most up to date status of the event. Devices can also subscribe to a specific resource instance to
 5133 get updates when any of its attributes change, including the Status object.

5134 ***currentStatus attribute (UInt8)***

5138 Field representing the current status type.

5139 0 = Scheduled

5140 This status indicates that the event has been scheduled and the event has not yet started. The server
5141 SHALL set the event to this status when the event is first scheduled and persist until the event has become
5142 active or has been cancelled. For events with a start time less than or equal to the current time, this status
5143 SHALL never be indicated, the event SHALL start with a status of "Active".

5144 1 = Active

5145 This status indicates that the event is currently active. The server SHALL set the event to this status when
5146 the event reaches its earliest Effective Start Time.

5147 2 = Cancelled

5148 When events are cancelled, the Status.dateTime attribute SHALL be set to the time the cancellation
5149 occurred, which cannot be in the future. The server is responsible for maintaining the cancelled event in
5150 its collection for the duration of the original event, or until the server has run out of space and needs to
5151 store a new event. Client devices SHALL be aware of Cancelled events, determine if the Cancelled event
5152 applies to them, and cancel the event immediately if applicable.

5153 3 = Cancelled with Randomization

5154 The server is responsible for maintaining the cancelled event in its collection for the duration of the
5155 Effective Scheduled Period. Client devices SHALL be aware of Cancelled with Randomization events,
5156 determine if the Cancelled event applies to them, and cancel the event immediately, using the larger of
5157 (absolute value of randomizeStart) and (absolute value of randomizeDuration) as the end randomization,
5158 in seconds. This Status.type SHALL NOT be used with "regular" Events, only with specializations of
5159 RandomizableEvent.

5160 4 = Superseded

5161 Events marked as Superseded by servers are events that may have been replaced by new events that target
5162 the same group of device types and overlap for a given period of time. Servers SHALL mark an event as
5163 Superseded at the earliest Effective Start Time of the overlapping event. Servers are responsible for
5164 maintaining the Superseded event in their collection for the duration of the Effective Scheduled Period.

5165 Client devices encountering a Superseded event SHALL terminate execution of the event immediately
5166 and commence execution of the new event immediately, unless the current time is within the start
5167 randomization window of the superseded event, in which case the client SHALL obey the start
5168 randomization of the new event. This Status.type SHALL NOT be used with TextMessage, since multiple
5169 text messages can be active.

5170 All other values reserved.

dateTime attribute (TimeType)

5171 The dateTime attribute will provide a timestamp of when the current status was defined. dateTime MUST
5172 be set to the time at which the status change occurred, not a time in the future or past.

potentiallySuperseded attribute (boolean)

5173 Set to true by a server of this event if there are events that may overlap this event in time and also overlap
5174 in DeviceCategory on the same function set instance. SHALL NOT be set to true if the event is a
5175 TextMessage.

5178 ***potentiallySupersededTime attribute (TimeType) [0..1]***

5179 Indicates the time that the potentiallySuperseded flag was set.

5180 ***reason attribute (String192) [0..1]***

5181 The Reason attribute allows a Service provider to provide a textual explanation of the status.

5182 ***RandomizableEvent Object (Event)***

5183 An Event that can indicate time ranges over which the start time and duration SHALL be
5184 randomized.

5185 ***randomizeDuration attribute (OneHourRangeType) [0..1]***

5186 Number of seconds boundary inside which a random value must be selected to be applied to the
5187 associated interval duration, to avoid sudden synchronized demand changes. If related to price level
5188 changes, sign may be ignored. Valid range is -3600 to 3600. If not specified, 0 is the default.

5189 ***randomizeStart attribute (OneHourRangeType) [0..1]***

5190 Number of seconds boundary inside which a random value must be selected to be applied to the
5191 associated interval start time, to avoid sudden synchronized demand changes. If related to price level
5192 changes, sign may be ignored. Valid range is -3600 to 3600. If not specified, 0 is the default.

5193 **15.1.2.3 Types Package**

5194 Contains definitions of reusable data types.

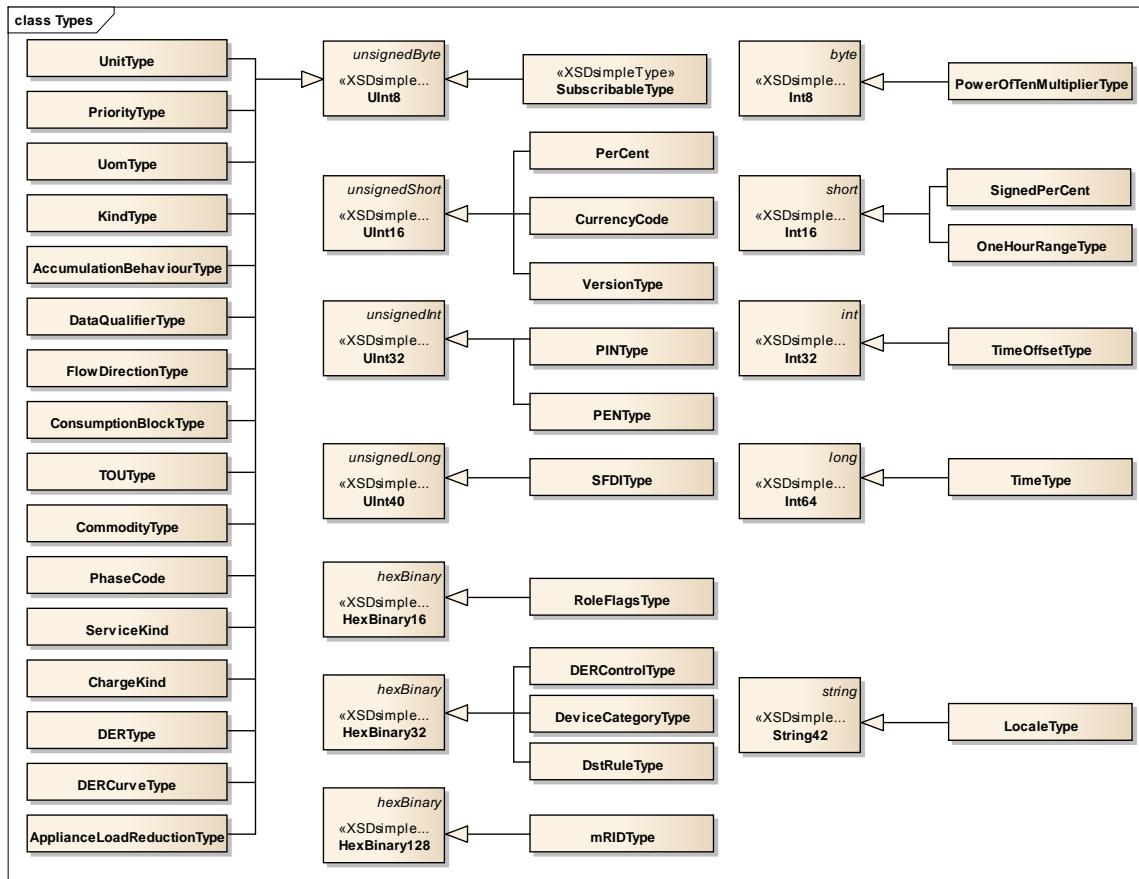


Figure 15-7: Types

5197 **AccumulationBehaviourType Object** (UInt8)

5198 0 = Not Applicable (default, if not specified)

5199 3 = Cumulative

5200 The sum of the previous billing period values. Note: “Cumulative” is commonly used in
 5201 conjunction with “demand.” Each demand reset causes the maximum demand value for the
 5202 present billing period (since the last demand reset) to accumulate as an accumulative total of all
 5203 maximum demands. So instead of “zeroing” the demand register, a demand reset has the affect of
 5204 adding the present maximum demand to this accumulating total.

5205 4 = DeltaData

5206 The difference between the value at the end of the prescribed interval and the beginning of the
 5207 interval. This is used for incremental interval data.

5208 Note: One common application would be for load profile data, another use might be to report the
 5209 number of events within an interval (such as the number of equipment energizations within the
 5210 specified period of time.)

5211 6 = Indicating

5212 As if a needle is swung out on the meter face to a value to indicate the current value. (Note: An
 5213 “indicating” value is typically measured over hundreds of milliseconds or greater, or may imply a
 5214 “pusher” mechanism to capture a value. Compare this to “instantaneous” which is measured over
 5215 a shorter period of time.)

5216 9 = Summation

5217 A form of accumulation which is selective with respect to time.

5218 Note : “Summation” could be considered a specialization of “Bulk Quantity” according to the
 5219 rules of inheritance where “Summation” selectively accumulates pulses over a timing pattern, and
 5220 “BulkQuantity” accumulates pulses all of the time.

5221 12 = Instantaneous

5222 Typically measured over the fastest period of time allowed by the definition of the metric (usually
 5223 milliseconds or tens of milliseconds.) (Note: “Instantaneous” was moved to attribute #3 in 61968-
 5224 9Ed2 from attribute #1 in 61968-9Ed1.)

5225 All other values reserved.

5226 **ApplianceLoadReductionType Object** (UInt8)

5227 0 - Delay Appliance Load

5228 Parameter requesting the appliance to respond by providing a moderate load reduction for the
 5229 duration of a delay period. Typically referring to a “non-emergency” event in which appliances
 5230 can continue operating if already in a load consuming period.

5231 1 - Temporary Appliance Load Reduction

5232 Parameter requesting the appliance to respond by providing an aggressive load reduction for a
 5233 short time period. Typically referring to an “emergency/spinning reserve” event in which an
 5234 appliance should start shedding load if currently in a load consuming period.

5235 * Full definition of how appliances react when receiving each parameter is document in the EPA
5236 document - ENERGY STAR® Program Requirements, Product Specification for Residential
5237 Refrigerators and Freezers, Eligibility Criteria 5, Draft 2 Version 5.0.

5238 All other values reserved.

CommodityType Object (UInt8)

5239 0 = Not Applicable (default, if not specified)

5241 1 = Electricity secondary metered value (a premises meter is typically a secondary meter)

5242 2 = Electricity primary metered value

5243 4 = Air

5244 7 = NaturalGas

5245 8 = Propane

5246 9 = PotableWater

5247 10 = Steam

5248 11 = WasteWater

5249 12 = HeatingFluid

5250 13 = CoolingFluid

5251 All other values reserved.

ConsumptionBlockType Object (UInt8)

5252 0 = Not Applicable (default, if not specified)

5254 1 = Block 1

5255 2 = Block 2

5256 3 = Block 3

5257 4 = Block 4

5258 5 = Block 5

5259 6 = Block 6

5260 7 = Block 7

5261 8 = Block 8

5262 9 = Block 9

5263 10 = Block 10

5264 11 = Block 11

5265 12 = Block 12

5266 13 = Block 13

5267 14 = Block 14

5268 15 = Block 15

- 5269 16 = Block 16
- 5270 All other values reserved.
- 5271 **CurrencyCode Object** (UInt16)
- 5272 Follows codes defined in [ISO 4217].
- 5273 0 - Not Applicable (default, if not specified)
- 5274 36 - Australian Dollar
- 5275 124 - Canadian Dollar
- 5276 840 - US Dollar
- 5277 978 - Euro
- 5278 This is not a complete list.
- 5279 **DataQualifierType Object** (UInt8)
- 5280 0 = Not Applicable (default, if not specified)
- 5281 2 = Average
- 5282 8 = Maximum
- 5283 9 = Minimum
- 5284 12 = Normal
- 5285 All other values reserved.
- 5286 **DateTimeInterval Object** «Compound» ()
- 5287 Interval of date and time.
- 5288 **duration attribute** (UInt32)
- 5289 Duration of the interval, in seconds.
- 5290 **start attribute** (TimeType)
- 5291 Date and time of the start of the interval.
- 5292 **DeviceCategoryType Object** (HexBinary32)
- 5293 The Device category types defined.
- 5294 Bit positions SHALL be defined as follows:
- 5295 0 - Programmable Communicating Thermostat
- 5296 1 - Strip Heaters
- 5297 2 - Baseboard Heaters
- 5298 3 - Water Heater
- 5299 4 - Pool Pump
- 5300 5 - Sauna
- 5301 6 - Hot tub
- 5302 7 - Smart Appliance

5303 8 - Irrigation Pump
5304 9 - Managed Commercial and Industrial (C&I) Loads
5305 10 - Simple misc. (Residential On/Off) loads
5306 11 - Exterior Lighting
5307 12 - Interior Lighting
5308 13 - Electric Vehicle
5309 14 - Generation Systems
5310 15 - Load Control Switch
5311 16 - Smart Inverter
5312 17 - EVSE
5313 18 - RESU
5314 19 - Energy Management System
5315 20 - Smart Energy Module
5316 All other values reserved.

5317 DstRuleType Object (HexBinary32)

5318 Bit map encoded rule from which is calculated the start or end time, within the current year, to
5319 which daylight savings time offset must be applied.

5320 The rule encoding:

5321 Bits 0 - 11: seconds 0 - 3599

5322 Bits 12 - 16: hours 0 - 23

5323 Bits 17 - 19: day of the week 0 = not applicable, 1 - 7 (Monday = 1)

5324 Bits:20 - 24: day of the month 0 = not applicable, 1 - 31

5325 Bits: 25 - 27: operator (detailed below)

5326 Bits: 28 - 31: month 1 - 12

5327 Rule value of 0xFFFFFFFF means rule processing/DST correction is disabled.

5328 The operators:

5329 0: DST starts/ends on the Day of the Month

5330 1: DST starts/ends on the Day of the Week that is on or after the Day of the Month

5331 2: DST starts/ends on the first occurrence of the Day of the Week in a month

5332 3: DST starts/ends on the second occurrence of the Day of the Week in a month

5333 4: DST starts/ends on the third occurrence of the Day of the Week in a month

5334 5: DST starts/ends on the forth occurrence of the Day of the Week in a month

5335 6: DST starts/ends on the fifth occurrence of the Day of the Week in a month

5336 7: DST starts/ends on the last occurrence of the Day of the Week in a month

5337 An example: DST starts on third Friday in March at 1:45 AM. The rule...

5338 Seconds: 2700

5339 Hours: 1

5340 Day of Week: 5

5341 Day of Month: 0

5342 Operator: 4

5343 Month: 3

5344 FlowDirectionType Object (UInt8)

5345 0 = Not Applicable (default, if not specified)

5346 1 = Forward (delivered to customer)

5347 19 = Reverse (received from customer)

5348 All other values reserved.

5349 KindType Object (UInt8)

5350 0 = Not Applicable (default, if not specified)

5351 3 = Currency

5352 8 = Demand

5353 12 = Energy

5354 37 = Power

5355 All other values reserved.

5356 LocaleType Object (String42)

5357 [RFC 4646] identifier of a language-region

5358 mRIDType Object (HexBinary128)

5359 A master resource identifier. The IANA PEN [PEN] provider ID SHALL be specified in bits 0-
5360 31, the least-significant bits, and objects created by that provider SHALL be assigned unique IDs
5361 with the remaining 96 bits.

5362 0xFFFFFFFFFFFFFFFFFFFF[XXXXXXXX], where [XXXXXXXX] is the PEN, is
5363 reserved for a object that is being created (e.g., a ReadingSet for the current time that is still
5364 accumulating).

5365 Except for this special reserved identifier, each modification of an object (resource)
5366 representation MUST have a different "version".

5367 OneHourRangeType Object (Int16)

5368 A signed time offset, typically applied to a Time value, expressed in seconds, with range -3600 to
5369 3600.

5370 PENType Object (UInt32)

5371 IANA Private Enterprise Number [PEN].

5372 **PerCent Object** (UInt16)

5373 Used for percentages, specified in hundredths of a percent, 0 - 10000. (10000 = 100%)

5374 **PhaseCode Object** (UInt8)

5375 0 = Not Applicable (default, if not specified)

5376 32 = Phase C (and S2)

5377 33 = Phase CN (and S2N)

5378 40 = Phase CA

5379 64 = Phase B

5380 65 = Phase BN

5381 66 = Phase BC

5382 128 = Phase A (and S1)

5383 129 = Phase AN (and S1N)

5384 132 = Phase AB

5385 224 = Phase ABC

5386 All other values reserved.

5387 **PINType Object** (UInt32)

5388 6 digit unsigned decimal integer (0 - 999999).

5389 (Note that this only requires 20 bits, if it can be allocated.)

5390 **PowerOfTenMultiplierType Object** (Int8)

5391 -9 = nano=x10^-9

5392 -6 = micro=x10^-6

5393 -3 = milli=x10^-3

5394 0 = none=x1 (default, if not specified)

5395 1 = deca=x10

5396 2 = hecto=x100

5397 3 = kilo=x1000

5398 6 = Mega=x10^6

5399 9 = Giga=x10^9

5400 This is not a complete list. Any integer between -9 and 9 SHALL be supported, indicating the
5401 power of ten multiplier for the units.

5402 **PrimacyType Object** (UInt8)

5403 Values possible for indication of "Primary" provider:

5404 0: In home energy management system

5405 1: Contracted premises service provider

5406 2: Non-contractual service provider

5407 All other values reserved.

RealEnergy Object ()

5409 Real electrical energy

5410 **multiplier attribute** (*PowerOfTenMultiplierType*)

5411 Multiplier for 'unit'.

5412 **value attribute** (*UInt48*)

5413 Value of the energy in Watt-hours. (uom 72)

RoleFlagsType Object (*HexBinary16*)

5415 Specifies the roles that apply to a usage point.

5416 Bit 0 - isMirror - SHALL be set if the server is not the measurement device

5417 Bit 1 - isPremisesAggregationPoint - SHALL be set if the UsagePoint is the point of delivery for
5418 a premises

5419 Bit 2 - isPEV - SHALL be set if the usage applies to an electric vehicle

5420 Bit 3 - isDER - SHALL be set if the usage applies to a distributed energy resource, capable of
5421 delivering power to the grid.

5422 Bit 4 - isRevenueQuality - SHALL be set if usage was measured by a device certified as revenue
5423 quality

5424 Bit 5 - isDC - SHALL be set if the usage point measures direct current

5425 Bit 6 - isSubmeter - SHALL be set if the usage point is not a premises aggregation point

5426 Bit 7-15 - Reserved

ServiceKind Object (*UInt8*)

5428 Service kind

5429 0 - electricity

5430 1 - gas

5431 2 - water

5432 3 - time

5433 4 - pressure

5434 5 - heat

5435 6 - cooling

5436 All other values reserved.

SFDIType Object (*UInt40*)

5438 Unsigned integer, max inclusive 687194767359, which is $2^{36}-1$ (68719476735), with added
5439 check digit. See Section 8.3.2 for check digit calculation.

SignedPerCent Object (*Int16*)

5441 Used for signed percentages, specified in hundredths of a percent, -10000 - 10000. (10000 =

5442 100%)

SignedRealEnergy Object ()

5444 Real electrical energy, signed.

5445 **multiplier attribute** (*PowerOfTenMultiplierType*)

5446 Multiplier for 'unit'.

5447 **value attribute** (*Int48*)

5448 Value of the energy in Watt-hours. (uom 72)

SubscribableType Object «XSDsimpleType» (UInt8)

5449 The subscribable values.

5451 0 - Resource does not support subscriptions

5452 1 - Resource supports non-conditional subscriptions

5453 2 - Resource supports conditional subscriptions

5454 3 - Resource supports both conditional and non-conditional subscriptions

5455 All other values reserved.

TimeOffsetType Object (Int32)

5457 A signed time offset, typically applied to a Time value, expressed in seconds.

TimeType Object (Int64)

5458 Time is a signed 64 bit value representing the number of seconds since 0 hours, 0 minutes, 0 seconds, on the 1st of January, 1970, in UTC, not counting leap seconds.

TOUType Object (UInt8)

5462 0 = Not Applicable (default, if not specified)

5463 1 = TOU A

5464 2 = TOU B

5465 3 = TOU C

5466 4 = TOU D

5467 5 = TOU E

5468 6 = TOU F

5469 7 = TOU G

5470 8 = TOU H

5471 9 = TOU I

5472 10 = TOU J

5473 11 = TOU K

5474 12 = TOU L

5475 13 = TOU M

5476 14 = TOU N

5477 15 = TOU O

5478 All other values reserved.

UnitType Object (UInt8)

5480 The unit types defined for end device control target reductions.

5481 0 - kWh

5482 1 - kW

5483 2 - Watts

5484 3 - Cubic Meters

5485 4 - Cubic Feet

5486 5 - US Gallons

5487 6 - Imperial Gallons

5488 7 - BTUs

5489 8 - Liters

5490 9 - kPa (gauge)

5491 10 - kPa (absolute)

5492 11 - Mega Joule

5493 12 - Unitless

5494 All other values reserved.

UnitValueType Object ()

5495 Type for specification of a specific value, with units and power of ten multiplier.

5496 **multiplier attribute (PowerOfTenMultiplierType)**

5497 Multiplier for 'unit'.

5498 **unit attribute (UomType)**

5499 Unit in symbol

5500 **value attribute (Int32)**

5501 Value in units specified

UomType Object (UInt8)

5502 0 = Not Applicable (default, if not specified)

5503 5 = A (Current in Amperes (RMS))

5504 6 = Kelvin (Temperature)

5505 23 = Degrees Celsius (Relative temperature)

5506 29 = Voltage

5507 31 = J (Energy joule)

5508 33 = Hz (Frequency)

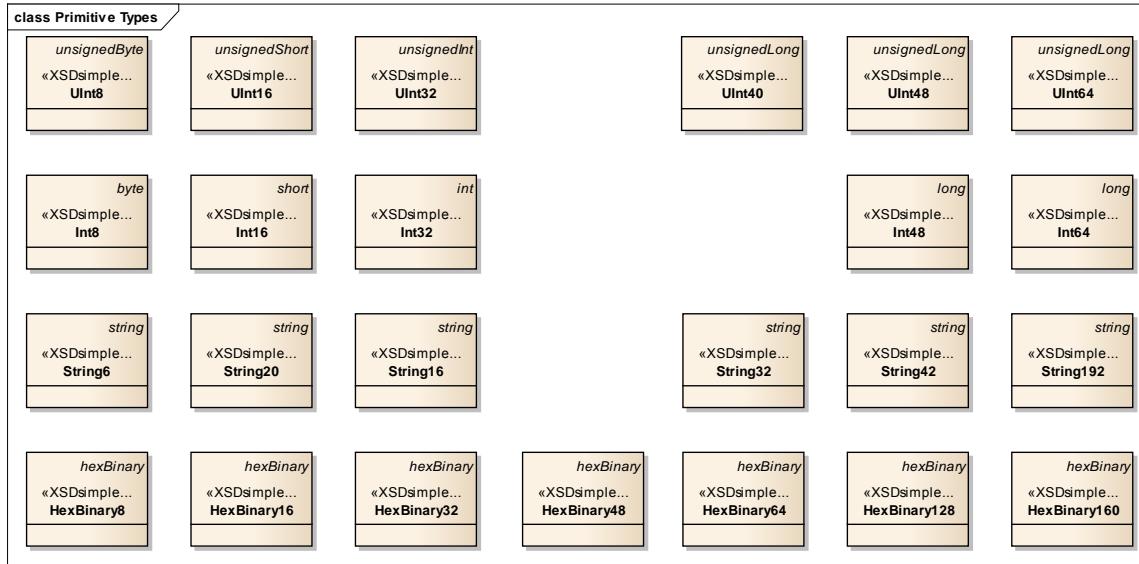
5511 38 =W (Real power in Watts)
5512 42 = m³ (Cubic Meter)
5513 61 = VA (Apparent power)
5514 63 = var (Reactive power)
5515 65 = CosTheta (Displacement Power Factor)
5516 67 = V² (Volts squared)
5517 69 = A² (Amp squared)
5518 71 = VAh (Apparent energy)
5519 72 = Wh (Real energy in Watt-hours)
5520 73 = varh (Reactive energy)
5521 106 = Ah (Ampere-hours / Available Charge)
5522 119 = ft³ (Cubic Feet)
5523 122 = ft³/h (Cubic Feet per Hour)
5524 125 = m³/h (Cubic Meter per Hour)
5525 128 = US gl (US Gallons)
5526 129 = US gl/h (US Gallons per Hour)
5527 130 = IMP gl (Imperial Gallons)
5528 131 = IMP gl/h (Imperial Gallons per Hour)
5529 132 = BTU
5530 133 = BTU/h
5531 134 = Liter
5532 137 = L/h (Liters per Hour)
5533 140 = PA(gauge)
5534 155 = PA(absolute)
5535 169 = Therm
5536 All other values reserved.

5537 **VersionType Object** (UInt16)

5538 Version SHALL indicate a distinct identifier for each revision of an IdentifiedObject. If not
5539 specified, a default version of "0" (initial version) SHALL be assumed. Upon modification of any
5540 IdentifiedObject, the mRID SHALL remain the same, but the version SHALL be incremented.
5541 Servers MAY NOT modify objects that they did not create, unless they were notified of the
5542 change from the entity controlling the object's PEN.

5543 **15.1.2.4 Primitive Types Package**

5544 Contains definitions of primitive data types based on XML schema primitives.



5545

5546 **Figure 15-8: Primitive Types**

5547 **HexBinary8 Object »XSDsimpleType» (hexBinary)**

5548 An 8-bit field encoded as a hex string (2 hex characters). Where applicable, bit 0, or the least
5549 significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd
5550 number of characters requires a leading "0".

5551 **HexBinary16 Object »XSDsimpleType» (hexBinary)**

5552 A 16-bit field encoded as a hex string (4 hex characters max). Where applicable, bit 0, or the least
5553 significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd
5554 number of characters requires a leading "0".

5555 **HexBinary32 Object »XSDsimpleType» (hexBinary)**

5556 A 32-bit field encoded as a hex string (8 hex characters max). Where applicable, bit 0, or the least
5557 significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd
5558 number of characters requires a leading "0".

5559 **HexBinary48 Object »XSDsimpleType» (hexBinary)**

5560 A 48-bit field encoded as a hex string (12 hex characters max). Where applicable, bit 0, or the least
5561 significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd
5562 number of characters requires a leading "0".

5563 **HexBinary64 Object »XSDsimpleType» (hexBinary)**

5564 A 64-bit field encoded as a hex string (16 hex characters max). Where applicable, bit 0, or the least
5565 significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd
5566 number of characters requires a leading "0".

5567 **HexBinary128 Object »XSDsimpleType» (hexBinary)**

5568 A 128-bit field encoded as a hex string (32 hex characters max). Where applicable, bit 0, or the least
5569 significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd
5570 number of characters requires a leading "0".

5571 HexBinary160 Object «XSDsimpleType» (hexBinary)

5572 A 160-bit field encoded as a hex string (40 hex characters max). Where applicable, bit 0, or the
5573 least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an
5574 odd number of characters requires a leading "0".

5575 String6 Object «XSDsimpleType» (string)

5576 Character string of max length 6. In order to limit internal storage, implementations SHALL
5577 reduce the length of strings using multi-byte characters so that the string may be stored using
5578 "maxLength" octets in the given encoding.

5579 String16 Object «XSDsimpleType» (string)

5580 Character string of max length 16. In order to limit internal storage, implementations SHALL
5581 reduce the length of strings using multi-byte characters so that the string may be stored using
5582 "maxLength" octets in the given encoding.

5583 String20 Object «XSDsimpleType» (string)

5584 Character string of max length 20. In order to limit internal storage, implementations SHALL
5585 reduce the length of strings using multi-byte characters so that the string may be stored using
5586 "maxLength" octets in the given encoding.

5587 String32 Object «XSDsimpleType» (string)

5588 Character string of max length 32. In order to limit internal storage, implementations SHALL
5589 reduce the length of strings using multi-byte characters so that the string may be stored using
5590 "maxLength" octets in the given encoding.

5591 String42 Object «XSDsimpleType» (string)

5592 Character string of max length 42. In order to limit internal storage, implementations SHALL
5593 reduce the length of strings using multi-byte characters so that the string may be stored using
5594 "maxLength" octets in the given encoding.

5595 String192 Object «XSDsimpleType» (string)

5596 Character string of max length 192. For all string types, in order to limit internal storage,
5597 implementations SHALL reduce the length of strings using multi-byte characters so that the
5598 string may be stored using "maxLength" octets in the given encoding.

5599 UInt8 Object «XSDsimpleType» (unsignedByte)

5600 Unsigned integer, max inclusive 255 (2^{8-1})

5601 UInt16 Object «XSDsimpleType» (unsignedShort)

5602 Unsigned integer, max inclusive 65535 (2^{16-1})

5603 UInt32 Object «XSDsimpleType» (unsignedInt)

5604 Unsigned integer, max inclusive 4294967295 (2^{32-1})

5605 UInt40 Object «XSDsimpleType» (unsignedLong)

5606 Unsigned integer, max inclusive 1099511627775 (2^{40-1})

5607 UInt48 Object «XSDsimpleType» (unsignedLong)

5608 Unsigned integer, max inclusive 281474976710655 (2^{48-1})

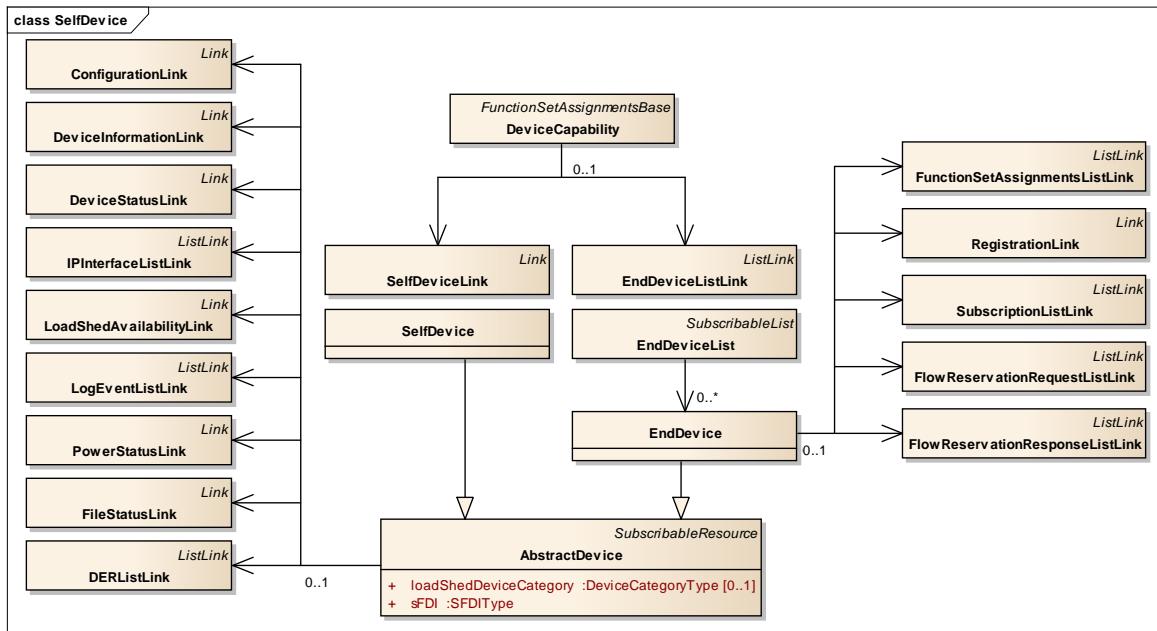
5609 UInt64 Object «XSDsimpleType» (unsignedLong)

5610 Unsigned integer, max inclusive 18446744073709551615 (2^{64-1})

- 5611 **Int8 Object** «XSDsimpleType» (byte)
 5612 Signed integer, min -128 max +127
- 5613 **Int16 Object** «XSDsimpleType» (short)
 5614 Signed integer, min -32768 max +32767
- 5615 **Int32 Object** «XSDsimpleType» (int)
 5616 Signed integer, max inclusive 2147483647 (2^{31}), min inclusive -2147483647 (same as xs:int)
- 5617 **Int48 Object** «XSDsimpleType» (long)
 5618 Signed integer, max inclusive 140737488355328 (2^{47}), min inclusive -140737488355328
- 5619 **Int64 Object** «XSDsimpleType» (long)
 5620 Signed integer, max inclusive 9223372036854775807 (2^{63}), min inclusive -
 5621 9223372036854775808 (same as xs:long)

5622 15.1.3 EndDevice Package

5623



5624

5625

Figure 15-9: SelfDevice

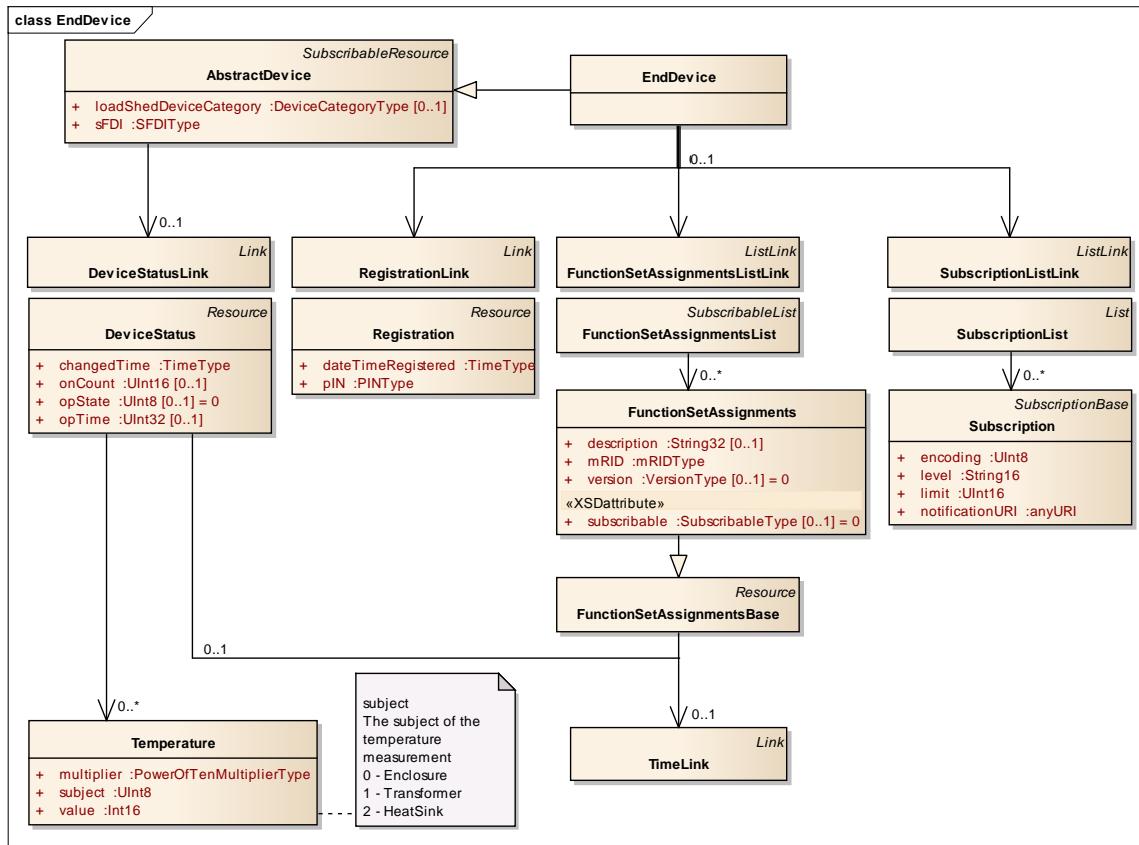


Figure 15-10: EndDevice

AbstractDevice Object (SubscribableResource)

5628 The EndDevice providing the resources available within the DeviceCapabilities.

loadShedDeviceCategory attribute (DeviceCategoryType) [0..1]

5630 This field is for use in devices that can shed load. If you are a device that does not respond to
5631 EndDeviceControls (for instance, an ESI), this field should not have any bits set.

sFDI attribute (SFDIType)

5632 Short form of device identifier, WITH the checksum digit. See the Security section for additional details.

DeviceStatus Object (Resource)

5633 Status of device

changedTime attribute (TimeType)

5634 The time at which the reported values were recorded.

onCount attribute (UInt16) [0..1]

5635 The number of times that the device has been turned on: Count of "device on" times, since the last time
5636 the counter was reset

opState attribute (UInt8) [0..1]

5637 Device operational state:

5638 0 - Not applicable / Unknown

5645 1 - Not operating
 5646 2 - Operating
 5647 3 - Starting up
 5648 4 - Shutting down
 5649 5 - At disconnect level
 5650 6 - kW ramping
 5651 7 - kVar ramping

opTime attribute (*UInt32*) [0..1]

5652 Total time device has operated: re-settable: Accumulated time in seconds since the last time the counter
 5653 was reset.

5655 **EndDevice Object** (*AbstractDevice*)

5656 Asset container that performs one or more end device functions. Contains information about
 5657 individual devices in the network.

5658 **EndDeviceList Object** (*SubscribableList*)

5659 A List element to hold EndDevice objects.

5660 **Registration Object** (*Resource*)

5661 Registration represents an authorization to access the resources on a host.

dateTimeRegistered attribute (*TimeType*)

5662 Contains the time at which this registration was created, by which clients MAY prioritize information
 5663 providers with the most recent registrations, when no additional direction from the consumer is available.

5665 ***pin attribute*** (*PINType*)

5666 Contains the registration PIN number associated with the device, including the checksum digit.

5667 **SelfDevice Object** (*AbstractDevice*)

5668 The EndDevice providing the resources available within the DeviceCapabilities.

5669 **Temperature Object** ()

5670 Specification of a temperature.

5671 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

5672 Multiplier for 'unit'.

5673 ***subject attribute*** (*UInt8*)

5674 The subject of the temperature measurement

5675 0 - Enclosure

5676 1 - Transformer

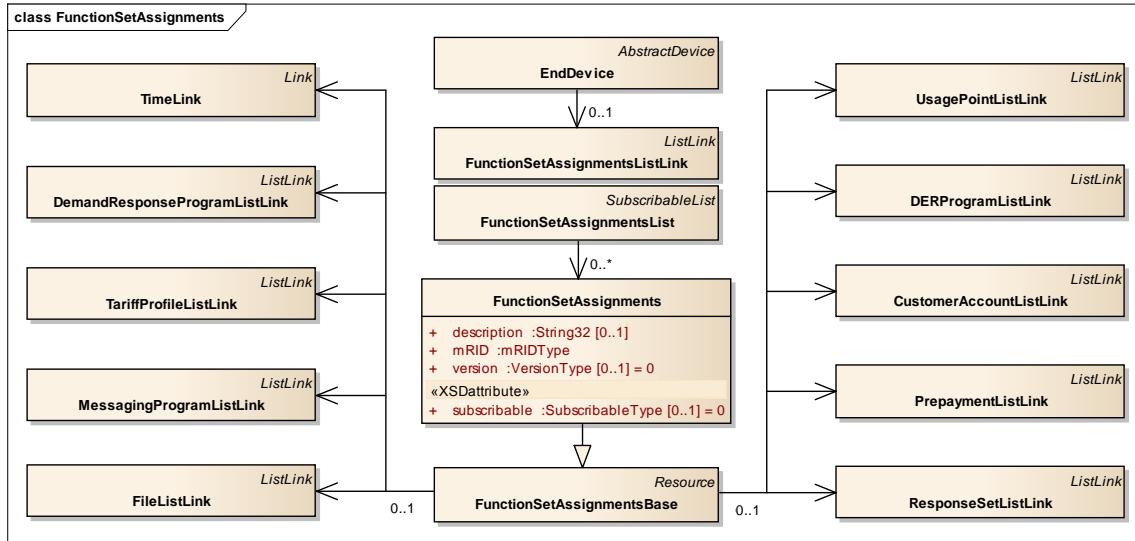
5677 2 - HeatSink

value attribute (*Int16*)

5678 Value in Degrees Celsius (uom 23).

5680 15.1.4 **FunctionSetAssignments Package**

5681



5682

5683

Figure 15-11: FunctionSetAssignments5684 **FunctionSetAssignmentsBase Object** (Resource)

5685 Defines a collection of function set instances that are to be used by one or more devices as
 5686 indicated by the EndDevice object(s) of the server.

5687 **FunctionSetAssignments Object** (FunctionSetAssignmentsBase)

5688 Provides an identifiable, subscribable collection of resources for a particular device to consume.

5689 **description attribute** (String32) [0..1]

5690 The description is a human readable text describing or naming the object.

5691 **mRID attribute** (mRIDType)

5692 The global identifier of the object.

5693 **subscribable attribute** (SubscribableType) [0..1] «XSDattribute»

5694 Indicates whether or not subscriptions are supported for this resource, and whether or not conditional
 5695 (thresholds) are supported. If not specified, is "not subscribable" (0).

5696 **version attribute** (VersionType) [0..1]

5697 Contains the version number of the object. See the type definition for details.

5698 **FunctionSetAssignmentsList Object** (SubscribableList)

5699 A List element to hold FunctionSetAssignments objects.

5700 15.1.5 **Pub-Sub Package**

5701 Contains resource definitions used to allow subscriptions and notifications of publications.

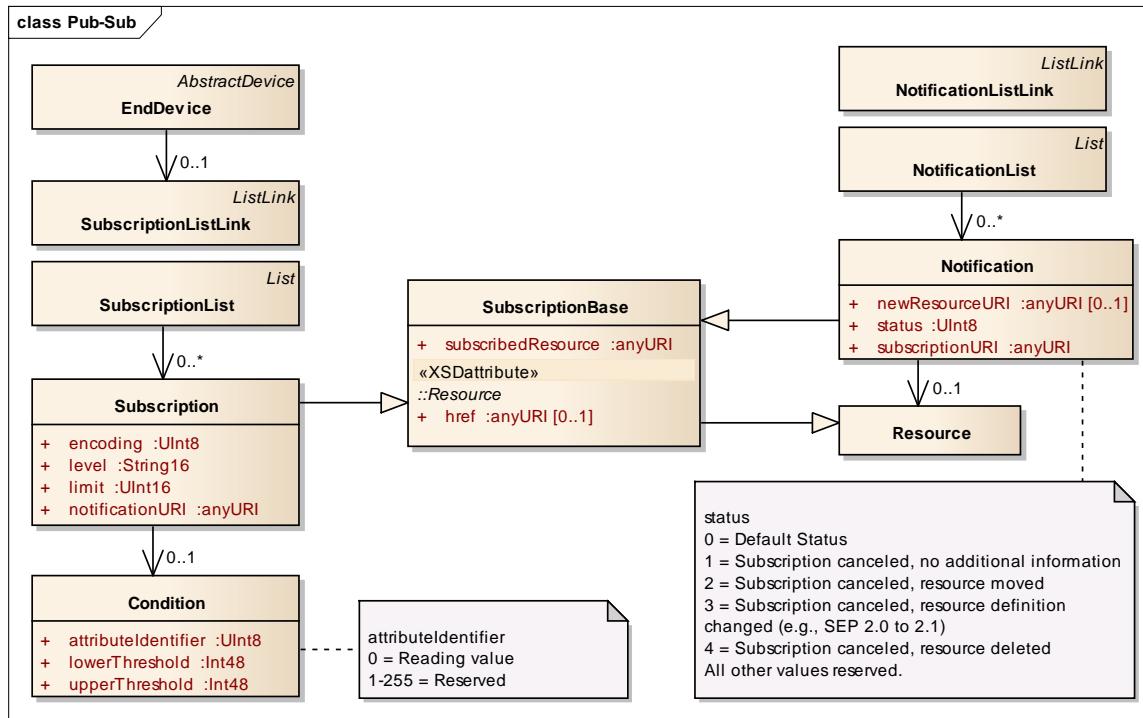


Figure 15-12: Pub-Sub

5702

5703 **Condition Object ()**

5704 Indicates a condition that must be satisfied for the Notification to be triggered.

5705 **attributelIdentifier attribute (UInt8)**

5706 0 = Reading value

5707 1-255 = Reserved

5708 **lowerThreshold attribute (Int48)**

5709 The value of the lower threshold

5710 **upperThreshold attribute (Int48)**

5711 The value of the upper threshold

5712 **SubscriptionBase Object (Resource)**

5713 Holds the information related to a client subscription to receive updates to a resource automatically. The actual resources may be passed in the Notification by specifying a specific xsi:type for the Resource and passing the full representation.

5714 **subscribedResource attribute (anyURI)**

5715 The resource for which the subscription applies. Query string parameters SHALL NOT be specified when subscribing to list resources. Should a query string parameter be specified, servers SHALL ignore them.

5716 **Subscription Object (SubscriptionBase)**

5717 Holds the information related to a client subscription to receive updates to a resource

5722 automatically.

5723 ***encoding attribute*** (*UInt8*)

5724 0 - application/sep+xml

5725 1 - application/sep-exi

5726 2-255 - reserved

5727 ***level attribute*** (*String16*)

5728 Contains the preferred schema and extensibility level indication such as "+S0"

5729 ***limit attribute*** (*UInt16*)

5730 This element is used to indicate the maximum number of list items that should be included in a
 5731 notification when the subscribed resource changes. This limit is meant to be functionally equivalent to the
 5732 'limit' query string parameter, but applies to both list resources as well as other resources. For list
 5733 resources, if a limit of '0' is specified, then notifications SHALL contain a list resource with results='0'
 5734 (equivalent to a simple change notification). For list resources, if a limit greater than '0' is specified, then
 5735 notifications SHALL contain a list resource with results equal to the limit specified (or less, should the
 5736 list contain fewer items than the limit specified or should the server be unable to provide the requested
 5737 number of items for any reason) and follow the same rules for list resources (e.g., ordering). For non-list
 5738 resources, if a limit of '0' is specified, then notifications SHALL NOT contain a resource representation
 5739 (equivalent to a simple change notification). For non-list resources, if a limit greater than '0' is specified,
 5740 then notifications SHALL contain the representation of the changed resource.

5741 ***notificationURI attribute*** (*anyURI*)

5742 The resource to which to post the notifications about the requested subscribed resource. Because this URI
 5743 will exist on a server other than the one being POSTed to, this attribute SHALL be a fully-qualified
 5744 absolute URI, not a relative reference.

5745 **SubscriptionList Object** (List)

5746 A List element to hold Subscription objects.

5747 **Notification Object** (SubscriptionBase)

5748 Holds the information related to a client subscription to receive updates to a resource
 5749 automatically. The actual resources may be passed in the Notification by specifying a specific
 5750 xsi:type for the Resource and passing the full representation.

5751 ***newResourceURI attribute*** (*anyURI*) [0..1]

5752 The new location of the resource, if moved.

5753 ***status attribute*** (*UInt8*)

5754 0 = Default Status

5755 1 = Subscription canceled, no additional information

5756 2 = Subscription canceled, resource moved

5757 3 = Subscription canceled, resource definition changed (e.g., SEP 2.0 to 2.1)

5758 4 = Subscription canceled, resource deleted

5759 All other values reserved.

5760 ***subscriptionURI attribute*** (*anyURI*)

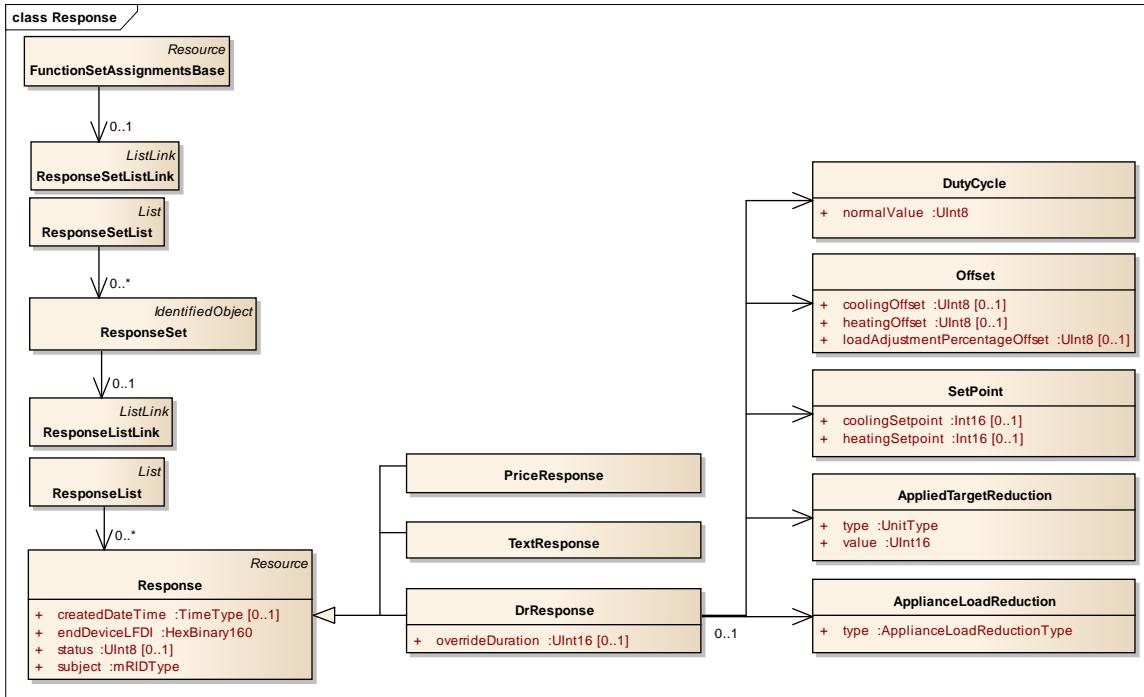
5761 The subscription from which this notification was triggered.

5762 **NotificationList Object** (List)

5763 A List element to hold Notification objects.

5764 **15.1.6 Response Package**

5765 Contains definitions of objects enabling responses to be sent back to suppliers and providers.



5766

Figure 15-13: Response

5768 **AppliedTargetReduction Object** ()

5769 Specifies the value of the TargetReduction applied by the device.

5770 **type attribute (UnitType)**

5771 Enumerated field representing the type of reduction requested.

5772 **value attribute (UInt16)**

5773 Indicates the requested amount of the relevant commodity to be reduced.

5774 **DrResponse Object** (Response)

5775 A response to a Demand Response Load Control (EndDeviceControl) message.

5776 **overrideDuration attribute (UInt16) [0..1]**

5777 Indicates the amount of time, in seconds, that the client partially opts-out during the demand response event. When overriding within the allowed override duration, the client SHALL send a partial opt-out (Response status code 8) for partial opt-out upon completion, with the total time the event was overridden (this attribute) populated. The client SHALL send a no participation status response (status type 10) if the user partially opts-out for longer than EndDeviceControl.overrideDuration.

5782 **PriceResponse Object** (Response)

5783 A response related to a price message.

5784 **Response Object** (Resource)

5785 The Response object is the generic response data repository for functions which do not have
5786 additional specific data (e.g. DRLC has additional data fields (SetPoint) where Price and Text
5787 event do not).

5788 **createdDateTime attribute** (TimeType) [0..1]

5789 The createdDateTime field contains the date and time when the acknowledgement/status occurred in the
5790 client. The client will provide the timestamp to ensure the proper time is captured in case the response is
5791 delayed in reaching the server (server receipt time would not be the same as the actual confirmation time).
5792 The time reported from the client should be relative to the time server indicated by the
5793 FunctionSetAssignment that also indicated the event resource; if no FunctionSetAssignment exists, the
5794 time of the server where the event resource was hosted.

5795 **endDeviceLFDI attribute** (HexBinary160)

5796 Contains the LFDI of the device providing the response.

5797 **status attribute** (UInt8) [0..1]

5798 The status field contains the acknowledgement or status. Each event type (DR/LC, Price, or Text) can
5799 return different status information (e.g. an Acknowledge will be returned for a Price event where a DRLC
5800 event can return Event Received, Event Started, and Event Completed). The Status field value definitions
5801 are defined in Table 10-10 Response Types by Function Set.

5802 **subject attribute** (mRIDType)

5803 The subject field provides a method to match the response with the originating event. It is populated with
5804 the mRID of the original object.

5805 **ResponseList Object** (List)

5806 A List element to hold Response objects.

5807 **ResponseSet Object** (IdentifiedObject)

5808 A container for a ResponseList.

5809 **ResponseSetList Object** (List)

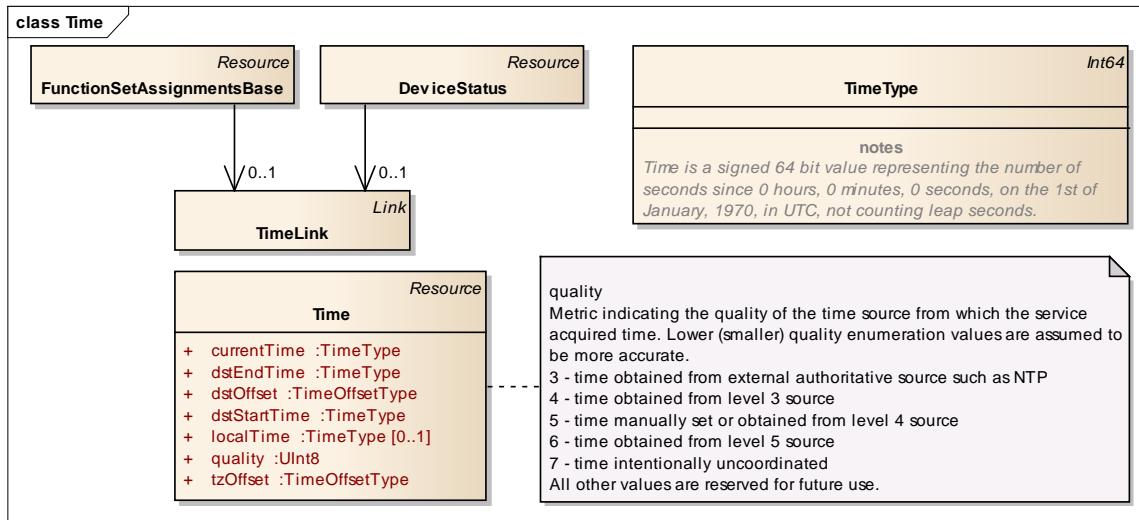
5810 A List element to hold ResponseSet objects.

5811 **TextResponse Object** (Response)

5812 A response to a text message

5813 15.1.7 **Time Package**

5814



5815

Figure 15-14: Time5817 **Time Object** (Resource)

5818 Contains the representation of time, constantly updated.

5819 **currentTime attribute** (TimeType)

5820 The current time, in the format defined by TimeType.

5821 **dstEndTime attribute** (TimeType)

5822 Time at which daylight savings ends (dstOffset no longer applied). Result of dstEndRule calculation.

5823 **dstOffset attribute** (TimeOffsetType)5824 Daylight savings time offset from local standard time. A typical practice is advancing clocks one hour
5825 when daylight savings time is in effect, which would result in a positive dstOffset.5826 **dstStartTime attribute** (TimeType)

5827 Time at which daylight savings begins (apply dstOffset). Result of dstStartRule calculation.

5828 **localTime attribute** (TimeType) [0..1]

5829 Local time: localTime = currentTime + tzOffset (+ dstOffset when in effect).

5830 **quality attribute** (UInt8)5831 Metric indicating the quality of the time source from which the service acquired time. Lower (smaller)
5832 quality enumeration values are assumed to be more accurate.

5833 3 - time obtained from external authoritative source such as NTP

5834 4 - time obtained from level 3 source

5835 5 - time manually set or obtained from level 4 source

5836 6 - time obtained from level 5 source

5837 7 - time intentionally uncoordinated

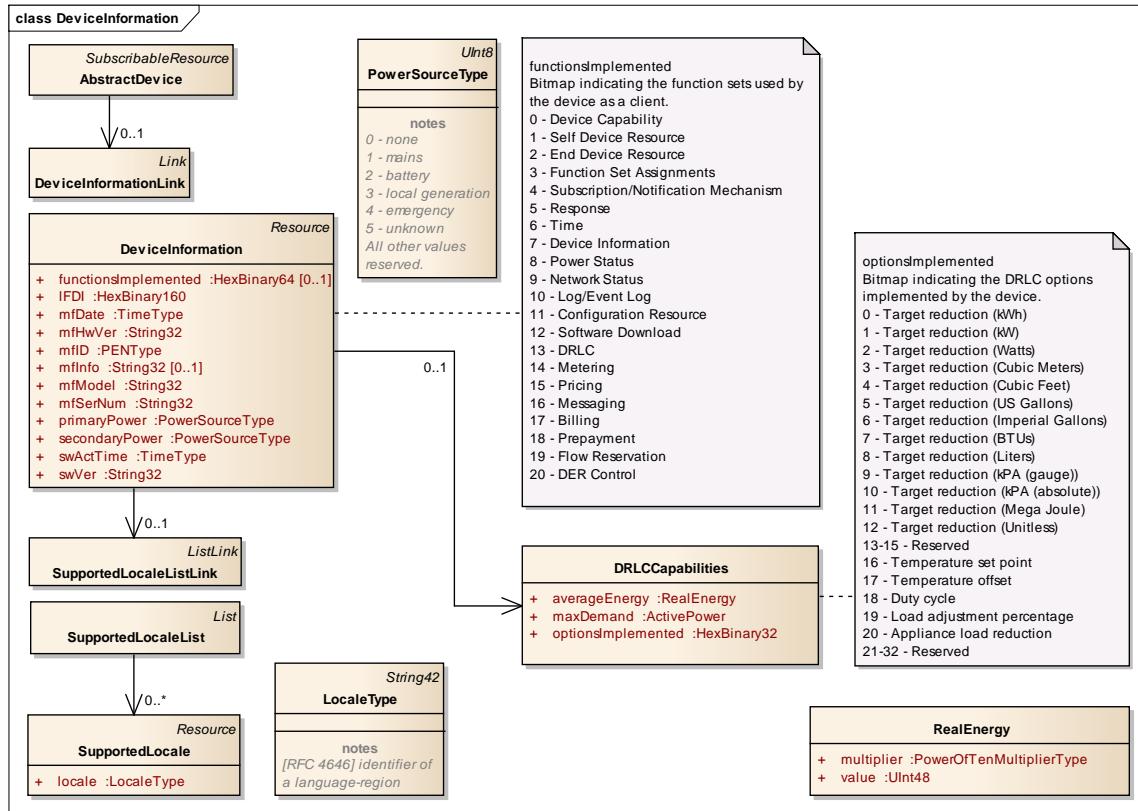
5838 All other values are reserved for future use.

5839 ***tzOffset attribute (TimeOffsetType)***

5840 Local time zone offset from currentTime. Does not include any daylight savings time offsets. For
 5841 American time zones, a negative tzOffset SHALL be used (eg, EST = GMT-5 which is -18000).

5842 **15.1.8 DeviceInformation Package**

5843 Contains general information about devices.



5844

5845 **Figure 15-15: DeviceInformation**

5846 **DeviceInformation Object (Resource)**

5847 Contains identification and other information about the device that changes very infrequently,
 5848 typically only when updates are applied, if ever.

5849 ***functionsImplemented attribute (HexBinary64) [0..1]***

5850 Bitmap indicating the function sets used by the device as a client.

5851 0 - Device Capability

5852 1 - Self Device Resource

5853 2 - End Device Resource

5854 3 - Function Set Assignments

5855 4 - Subscription/Notification Mechanism

5856 5 - Response

5857 6 - Time

5858 7 - Device Information
5859 8 - Power Status
5860 9 - Network Status
5861 10 - Log/Event Log
5862 11 - Configuration Resource
5863 12 - Software Download
5864 13 - DRLC
5865 14 - Metering
5866 15 - Pricing
5867 16 - Messaging
5868 17 - Billing
5869 18 - Prepayment
5870 19 - Flow Reservation
5871 20 - DER Control

5872 **IFDI attribute** (*HexBinary160*)

5873 Long form device identifier. See the Security section for full details.

5874 **mfDate attribute** (*TimeType*)

5875 Date/time of manufacture

5876 **mfHwVer attribute** (*String32*)

5877 Manufacturer hardware version

5878 **mfID attribute** (*PENType*)

5879 The manufacturer's IANA Enterprise Number.

5880 **mfInfo attribute** (*String32*) [0..1]

5881 Manufacturer dependent information related to the manufacture of this device

5882 **mfModel attribute** (*String32*)

5883 Manufacturer's model number

5884 **mfSerNum attribute** (*String32*)

5885 Manufacturer assigned serial number

5886 **primaryPower attribute** (*PowerSourceType*)

5887 Primary source of power.

5888 **secondaryPower attribute** (*PowerSourceType*)

5889 Secondary source of power

5890 **swActTime attribute** (*TimeType*)

5891 Activation date/time of currently running software

5892 **swVer attribute** (*String32*)

5893 Currently running software version

5894 **DRLCCapabilities Object ()**

5895 Contains information about the static capabilities of the device, to allow service providers to
5896 know what types of functions are supported, what the normal operating ranges and limits are, and
5897 other similar information, in order to provide better suggestions of applicable programs to receive
5898 the maximum benefit.

5899 **averageEnergy attribute (RealEnergy)**

5900 The average hourly energy usage when in normal operating mode.

5901 **maxDemand attribute (ActivePower)**

5902 The maximum demand rating of this end device.

5903 **optionsImplemented attribute (HexBinary32)**

5904 Bitmap indicating the DRLC options implemented by the device.

5905 0 - Target reduction (kWh)

5906 1 - Target reduction (kW)

5907 2 - Target reduction (Watts)

5908 3 - Target reduction (Cubic Meters)

5909 4 - Target reduction (Cubic Feet)

5910 5 - Target reduction (US Gallons)

5911 6 - Target reduction (Imperial Gallons)

5912 7 - Target reduction (BTUs)

5913 8 - Target reduction (Liters)

5914 9 - Target reduction (kPa (gauge))

5915 10 - Target reduction (kPa (absolute))

5916 11 - Target reduction (Mega Joule)

5917 12 - Target reduction (Unitless)

5918 13-15 - Reserved

5919 16 - Temperature set point

5920 17 - Temperature offset

5921 18 - Duty cycle

5922 19 - Load adjustment percentage

5923 20 - Appliance load reduction

5924 21-32 - Reserved

5925 **SupportedLocale Object (Resource)**

5926 Specifies a locale that is supported

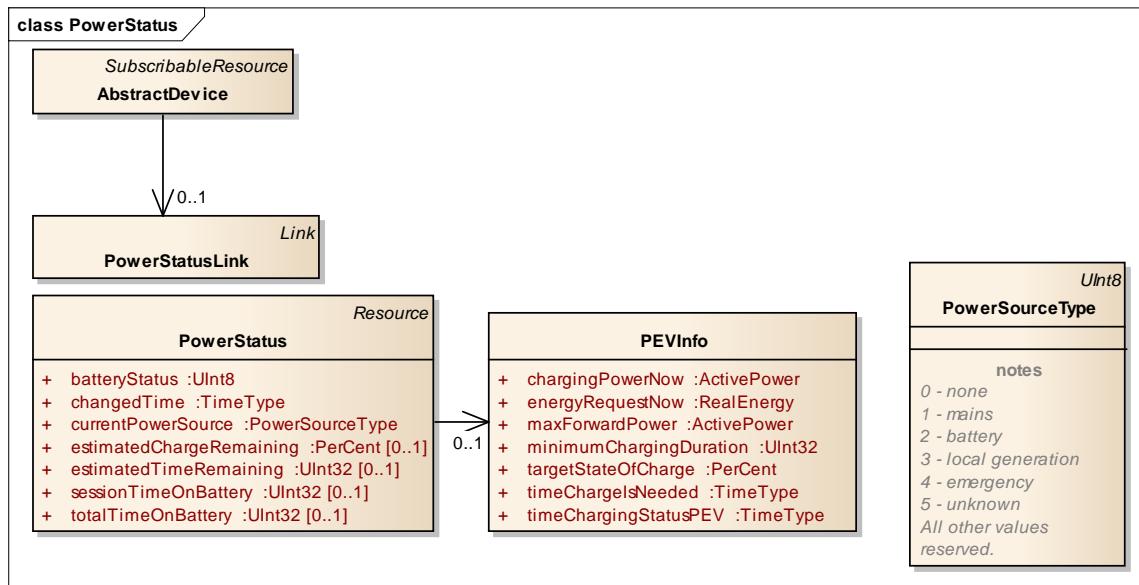
5927 **locale attribute (LocaleType)**

5928 The code for a locale that is supported

5929 **SupportedLocaleList Object** (List)
 5930 A List element to hold SupportedLocale objects.

5931 15.1.9 **PowerStatus Package**

5932



5933

5934 **Figure 15-16: PowerStatus**

5935 **PowerStatus Object** (Resource)
 5936 Contains the status of the device's power sources

5937 **batteryStatus attribute** (**UInt8**)

5938 Battery system status

5939 0 = unknown

5940 1 = normal (more than LowChargeThreshold remaining)

5941 2 = low (less than LowChargeThreshold remaining)

5942 3 = depleted (0% charge remaining)

5943 4 = not applicable (mains powered only)

5944 **changedTime attribute** (**TimeType**)

5945 The time at which the reported values were recorded.

5946 **currentPowerSource attribute** (**PowerSourceType**)

5947 This value will be fixed for devices powered by a single source. This value may change for devices able
 5948 to transition between multiple power sources (mains to battery backup, etc.).

5949 **estimatedChargeRemaining attribute** (**PerCent**) [0..1]

5950 Estimate of remaining battery charge as a percent of full charge.

5951 **estimatedTimeRemaining attribute** (**UInt32**) [0..1]

5952 Estimated time (in seconds) to total battery charge depletion (under current load)

5953 ***sessionTimeOnBattery attribute (UInt32) [0..1]***
 5954 If the device has a battery, this is the time since the device last switched to battery power, or the time
 5955 since the device was restarted, whichever is less, in seconds.

5956 ***totalTimeOnBattery attribute (UInt32) [0..1]***
 5957 If the device has a battery, this is the total time the device has been on battery power, in seconds. It may
 5958 be reset when the battery is replaced.

5959 **PowerSourceType Object (UInt8)**

- 5960 0 - none
 5961 1 - mains
 5962 2 - battery
 5963 3 - local generation
 5964 4 - emergency
 5965 5 - unknown
 5966 All other values reserved.

5967 **PEVInfo Object ()**

5968 Contains attributes that can be exposed by PEVs and other devices that have charging
 5969 requirements.

5970 ***chargingPowerNow attribute (ActivePower)***

5971 This is the actual power flow in or out of the charger or inverter. This is calculated by the vehicle based
 5972 on actual measurements. This number is positive for charging.

5973 ***energyRequestNow attribute (RealEnergy)***

5974 This is the amount of energy that must be transferred from the grid to EVSE and PEV to achieve the
 5975 target state of charge allowing for charger efficiency and any vehicle and EVSE parasitic loads. This is
 5976 calculated by the vehicle and changes throughout the connection as forward or reverse power flow change
 5977 the battery state of charge. This number is positive for charging.

5978 ***maxForwardPower attribute (ActivePower)***

5979 This is maximum power transfer capability that could be used for charging the PEV to perform the
 5980 requested energy transfer. It is the lower of the vehicle or EVSE physical power limitations. It is not
 5981 based on economic considerations. The vehicle may draw less power than this value based on its charging
 5982 cycle. The vehicle defines this parameter. This number is positive for charging power flow.

5983 ***minimumChargingDuration attribute (UInt32)***

5984 This is computed by the PEV based on the charging profile to complete the energy transfer if the
 5985 maximum power is authorized. The value will never be smaller than the ratio of the energy request to the
 5986 power request because the charging profile may not allow the maximum power to be used throughout the
 5987 transfer. This is a critical parameter for determining whether any slack time exists in the charging cycle
 5988 between the current time and the TCIN.

5989 ***targetStateOfCharge attribute (PerCent)***

5990 This is the target state of charge that is to be achieved during charging before the time of departure
 5991 (TCIN). The default value is 100%. The value cannot be set to a value less than the actual state of charge.

5992 ***timeChargelsNeeded attribute (TimeType)***

5993 Time Charge is Needed (TCIN) is the time that the PEV is expected to depart. The value is manually
 5994 entered using controls and displays in the vehicle or on the EVSE or using a mobile device. It is
 5995 authenticated and saved by the PEV. This value may be updated during a charging session.

timeChargingStatusPEV attribute (TimeType)

This is the time that the parameters are updated, except for changes to TCIN.

5998 15.1.10 **NetworkStatus Package**

5999

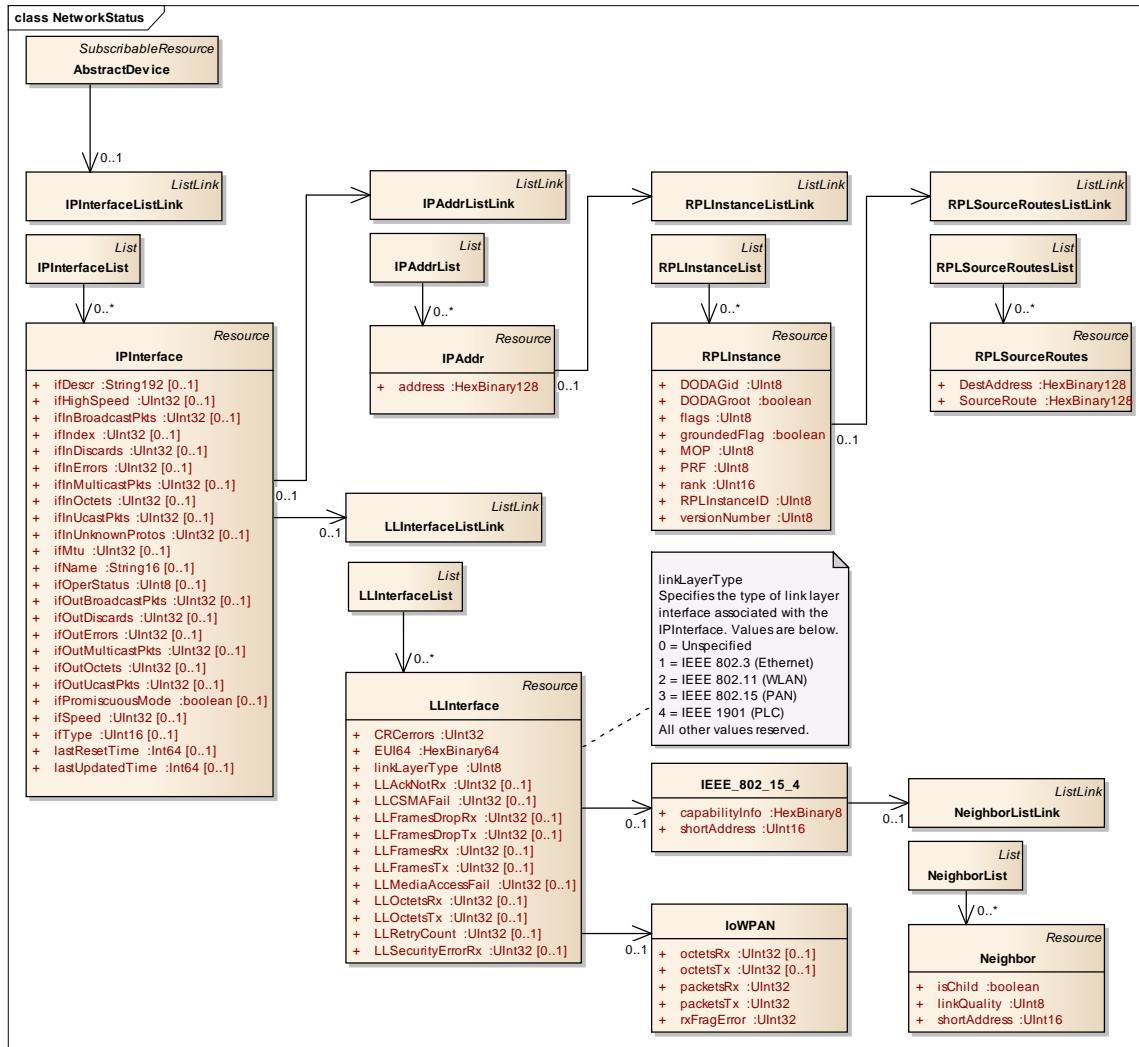


Figure 15-17: NetworkStatus

6000 6001 **IEEE_802_15_4 Object ()**

6002 Contains 802.15.4 link layer specific attributes.

6003 6004 **capabilityInfo attribute (HexBinary8)**

6005 As defined by IEEE 802.15.4

6006 **shortAddress attribute (UInt16)**

- 6007 As defined by IEEE 802.15.4
- 6008 **IPAddr Object** (Resource)
6009 An Internet Protocol address object.
- 6010 **address attribute** (*HexBinary128*)
6011 An IP address value.
- 6012 **IPAddrList Object** (List)
6013 List of IPAddr instances.
- 6014 **IPInterface Object** (Resource)
6015 Specific IPInterface resource. This resource may be thought of as network status information for
6016 a specific network (IP) layer interface.
- 6017 **ifDescr attribute** (*String192*) [0..1]
6018 Use rules from [RFC 2863].
- 6019 **ifHighSpeed attribute** (*UInt32*) [0..1]
6020 Use rules from [RFC 2863].
- 6021 **ifInBroadcastPkts attribute** (*UInt32*) [0..1]
6022 Use rules from [RFC 2863].
- 6023 **ifIndex attribute** (*UInt32*) [0..1]
6024 Use rules from [RFC 2863].
- 6025 **ifInDiscards attribute** (*UInt32*) [0..1]
6026 Use rules from [RFC 2863]. Can be thought of as Input Datagrams Discarded.
- 6027 **ifInErrors attribute** (*UInt32*) [0..1]
6028 Use rules from [RFC 2863].
- 6029 **ifInMulticastPkts attribute** (*UInt32*) [0..1]
6030 Use rules from [RFC 2863]. Can be thought of as Multicast Datagrams Received.
- 6031 **ifInOctets attribute** (*UInt32*) [0..1]
6032 Use rules from [RFC 2863]. Can be thought of as Bytes Received.
- 6033 **ifInUcastPkts attribute** (*UInt32*) [0..1]
6034 Use rules from [RFC 2863]. Can be thought of as Datagrams Received.
- 6035 **ifInUnknownProtos attribute** (*UInt32*) [0..1]
6036 Use rules from [RFC 2863]. Can be thought of as Datagrams with Unknown Protocol Received.
- 6037 **ifMtu attribute** (*UInt32*) [0..1]
6038 Use rules from [RFC 2863].
- 6039 **ifName attribute** (*String16*) [0..1]
6040 Use rules from [RFC 2863].
- 6041 **ifOperStatus attribute** (*UInt8*) [0..1]
6042 Use rules and assignments from [RFC 2863].
- 6043 **ifOutBroadcastPkts attribute** (*UInt32*) [0..1]
6044 Use rules from [RFC 2863]. Can be thought of as Broadcast Datagrams Sent.

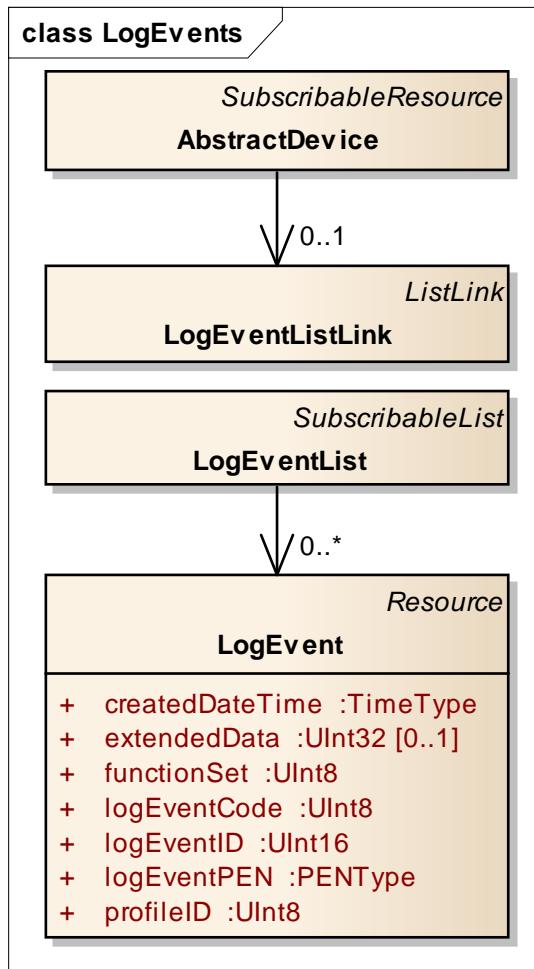
- 6045 ***ifOutDiscards attribute*** (*UInt32*) [0..1]
 6046 Use rules from [RFC 2863]. Can be thought of as Output Datagrams Discarded.
- 6047 ***ifOutErrors attribute*** (*UInt32*) [0..1]
 6048 Use rules from [RFC 2863].
- 6049 ***ifOutMulticastPkts attribute*** (*UInt32*) [0..1]
 6050 Use rules from [RFC 2863]. Can be thought of as Multicast Datagrams Sent.
- 6051 ***ifOutOctets attribute*** (*UInt32*) [0..1]
 6052 Use rules from [RFC 2863]. Can be thought of as Bytes Sent.
- 6053 ***ifOutUcastPkts attribute*** (*UInt32*) [0..1]
 6054 Use rules from [RFC 2863]. Can be thought of as Datagrams Sent.
- 6055 ***ifPromiscuousMode attribute*** (*boolean*) [0..1]
 6056 Use rules from [RFC 2863].
- 6057 ***ifSpeed attribute*** (*UInt32*) [0..1]
 6058 Use rules from [RFC 2863].
- 6059 ***ifType attribute*** (*UInt16*) [0..1]
 6060 Use rules and assignments from [RFC 2863].
- 6061 ***lastResetTime attribute*** (*Int64*) [0..1]
 6062 Similar to ifLastChange in [RFC 2863].
- 6063 ***lastUpdatedTime attribute*** (*Int64*) [0..1]
 6064 The date/time of the reported status.
- 6065 ***IPInterfaceList Object*** (*List*)
 6066 List of IPInterface instances.
- 6067 ***LLInterface Object*** (*Resource*)
 6068 A link-layer interface object.
- 6069 ***CRCerrors attribute*** (*UInt32*)
 6070 Contains the number of CRC errors since reset.
- 6071 ***EUI64 attribute*** (*HexBinary64*)
 6072 Contains the EUI-64 of the link layer interface. 48 bit MAC addresses SHALL be changed into an EUI-64
 6073 using the method defined in [RFC 4291], Appendix A. (The method is to insert "0xFFFF" as described in
 6074 the reference.)
- 6075 ***linkLayerType attribute*** (*UInt8*)
 6076 Specifies the type of link layer interface associated with the IPInterface. Values are below.
 6077 0 = Unspecified
 6078 1 = IEEE 802.3 (Ethernet)
 6079 2 = IEEE 802.11 (WLAN)
 6080 3 = IEEE 802.15 (PAN)
 6081 4 = IEEE 1901 (PLC)
 6082 All other values reserved.

- 6083 ***LLAckNotRx attribute*** (*UInt32*) [0..1]
 6084 Number of times an ACK was not received for a frame transmitted (when ACK was requested).
- 6085 ***LLCSMAFail attribute*** (*UInt32*) [0..1]
 6086 Number of times CSMA failed.
- 6087 ***LLFramesDropRx attribute*** (*UInt32*) [0..1]
 6088 Number of dropped receive frames.
- 6089 ***LLFramesDropTx attribute*** (*UInt32*) [0..1]
 6090 Number of dropped transmit frames.
- 6091 ***LLFramesRx attribute*** (*UInt32*) [0..1]
 6092 Number of link layer frames received.
- 6093 ***LLFramesTx attribute*** (*UInt32*) [0..1]
 6094 Number of link layer frames transmitted.
- 6095 ***LLMediaAccessFail attribute*** (*UInt32*) [0..1]
 6096 Number of times access to media failed.
- 6097 ***LLOctetsRx attribute*** (*UInt32*) [0..1]
 6098 Number of Bytes received.
- 6099 ***LLOctetsTx attribute*** (*UInt32*) [0..1]
 6100 Number of Bytes transmitted.
- 6101 ***LLRetryCount attribute*** (*UInt32*) [0..1]
 6102 Number of MAC transmit retries.
- 6103 ***LLSecurityErrorRx attribute*** (*UInt32*) [0..1]
 6104 Number of receive security errors.
- 6105 **LLInterfaceList Object** (List)
 6106 List of LLInterface instances.
- 6107 **IoWPAN Object** ()
 6108 Contains information specific to 6LoWPAN.
- 6109 ***octetsRx attribute*** (*UInt32*) [0..1]
 6110 Number of Bytes received
- 6111 ***octetsTx attribute*** (*UInt32*) [0..1]
 6112 Number of Bytes transmitted
- 6113 ***packetsRx attribute*** (*UInt32*)
 6114 Number of packets received
- 6115 ***packetsTx attribute*** (*UInt32*)
 6116 Number of packets transmitted
- 6117 ***rxFragError attribute*** (*UInt32*)
 6118 Number of errors receiving fragments
- 6119 **Neighbor Object** (Resource)
 6120 Contains 802.15.4 link layer specific attributes.

- 6121 ***isChild attribute*** (*boolean*)
 6122 True if the neighbor is a child.
- 6123 ***linkQuality attribute*** (*UInt8*)
 6124 The quality of the link, as defined by 802.15.4
- 6125 ***shortAddress attribute*** (*UInt16*)
 6126 As defined by IEEE 802.15.4
- 6127 **NeighborList Object** (List)
 6128 List of 15.4 neighbors.
- 6129 **RPLInstance Object** (Resource)
 6130 Specific RPLInstance resource. This resource may be thought of as network status information
 6131 for a specific RPL instance associated with IPInterface.
- 6132 ***DODAGid attribute*** (*UInt8*)
 6133 See [RFC 6550].
- 6134 ***DODAGroot attribute*** (*boolean*)
 6135 See [RFC 6550].
- 6136 ***flags attribute*** (*UInt8*)
 6137 See [RFC 6550].
- 6138 ***groundedFlag attribute*** (*boolean*)
 6139 See [RFC 6550].
- 6140 ***MOP attribute*** (*UInt8*)
 6141 See [RFC 6550].
- 6142 ***PRF attribute*** (*UInt8*)
 6143 See [RFC 6550].
- 6144 ***rank attribute*** (*UInt16*)
 6145 See [RFC 6550].
- 6146 ***RPLInstanceID attribute*** (*UInt8*)
 6147 See [RFC 6550].
- 6148 ***versionNumber attribute*** (*UInt8*)
 6149 See [RFC 6550].
- 6150 **RPLInstanceList Object** (List)
 6151 List of RPLInstances associated with the IPinterface.
- 6152 **RPLSourceRoutes Object** (Resource)
 6153 A RPL source routes object.
- 6154 ***DestAddress attribute*** (*HexBinary128*)
 6155 See [RFC 6554].
- 6156 ***SourceRoute attribute*** (*HexBinary128*)
 6157 See [RFC 6554].
- 6158 **RPLSourceRoutesList Object** (List)
 6159 List or RPL source routes if the hosting device is the DODAGroot

6160 15.1.11 **LogEvents Package**

6161



6162

6163

Figure 15-18: LogEvents6164 **LogEvent Object (Resource)**

6165 A time stamped instance of a significant event detected by the device.

6166 **createdDateTime attribute (TimeType)**

6167 The date and time that the event occurred.

6168 **extendedData attribute (UInt32) [0..1]**

6169 May be used to transmit additional details about the event.

6170 **functionSet attribute (UInt8)**

6171 If the profileID indicates this is the Smart Energy Profile, the functionSet is defined by the Zigbee
 6172 Alliance and SHALL be one of the values from the table below (Smart Energy Profile 2.0 function set
 6173 identifiers). If the profileID is anything else, the functionSet is defined by the identified profile.

6174 0 General (not specific to a function set)

6175 1 Publish and Subscribe

6176 2 End Device
 6177 3 Function Set Assignment
 6178 4 Response
 6179 5 Demand Response and Load Control
 6180 6 Metering
 6181 7 Pricing
 6182 8 Messaging
 6183 9 Billing
 6184 10 Prepayment
 6185 11 Distributed Energy Resources
 6186 12 Time
 6187 13 Software Download
 6188 14 Device Information
 6189 15 Power Status
 6190 16 Network Status
 6191 17 Log Event List
 6192 18 Configuration
 6193 19 Security
 6194 All other values are reserved.

logEventCode attribute (UInt8)

6195 An 8 bit unsigned integer. logEventCodes are scoped to a profile and a function set. If the profile is
 6196 Smart Energy, the logEventCode is defined by the Zigbee Alliance within one of the function sets of
 6197 Smart Energy Profile 2.0. If the profile is anything else, the logEventCode is defined by the specified
 6198 profile.
 6199

logEventID attribute (UInt16)

6200 This 16-bit value, combined with createdDateTime, profileID, and logEventPEN, should provide a
 6201 reasonable level of uniqueness.
 6202

logEventPEN attribute (PENType)

6203 The Private Enterprise Number(PEN) of the entity that defined the profileID, functionSet, and
 6204 logEventCode of the logEvent. ZigBee-assigned logEventCodes SHALL use the ZigBee Alliance PEN.
 6205 Combinations of profileID, functionSet, and logEventCode SHALL have unique meaning within a
 6206 logEventPEN and are defined by the owner of the PEN.
 6207

profileID attribute (UInt8)

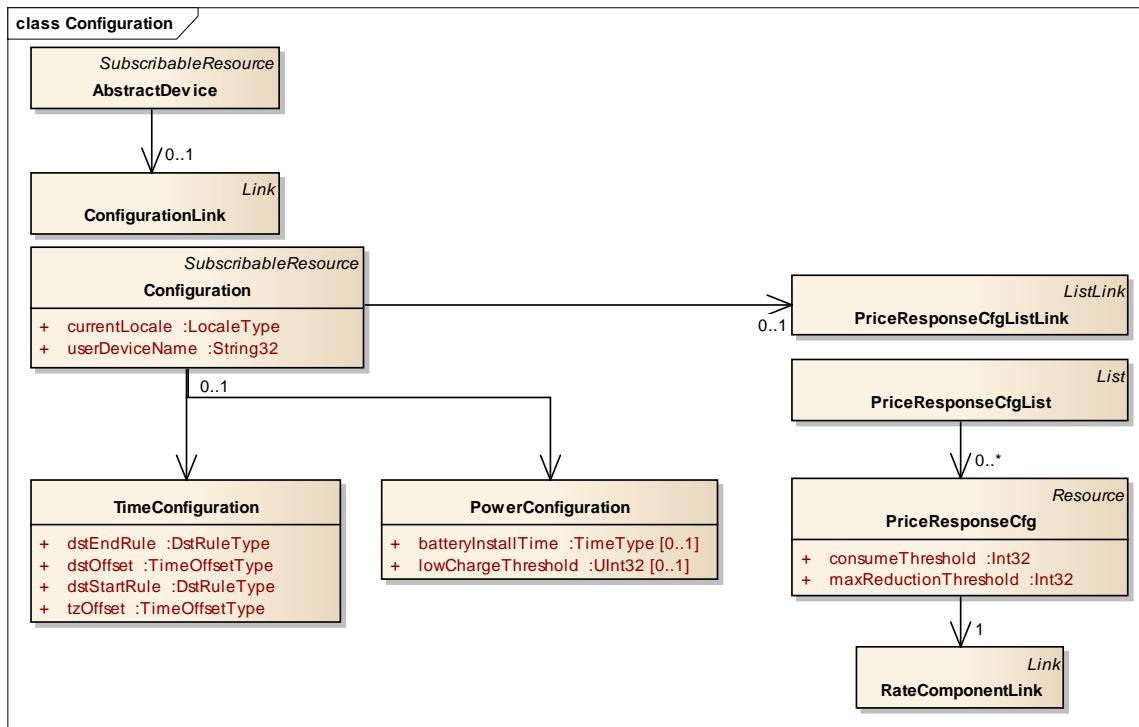
6208 The profileID identifies which profile (HA, BA, SE, etc) defines the following event information.
 6209
 6210 0 Not profile specific.
 6211 1 Vendor Defined

6212 2 Smart Energy
 6213 3 Home Automation
 6214 4 Building Automation
 6215 All other values are reserved.

6216 **LogEventList Object** (SubscribableList)
 6217 A List element to hold LogEvent objects.

6218 15.1.12 Configuration Package

6219



6220

6221 **Figure 15-19: Configuration**

6222 **Configuration Object** (SubscribableResource)

6223 This resource contains various settings to control the operation of the device

6224 **currentLocale attribute** (*LocaleType*)

6225 [RFC 4646] identifier of the language-region currently in use.

6226 **userDeviceName attribute** (*String32*)

6227 User assigned, convenience name used for network browsing displays, etc. Example "My Thermostat"

6228 **PowerConfiguration Object** ()

6229 Contains configuration related to the device's power sources

6230 **batteryInstallTime attribute** (*TimeType*) [0..1]

6231 Time/Date at which battery was installed,

6232 **lowChargeThreshold attribute** (*UInt32*) [0..1]

6233 In context of the PowerStatus resource, this is the value of EstimatedTimeRemaining below which
6234 BatteryStatus "low" is indicated and the LE_LOW_BATTERY is raised.

6235 **PriceResponseCfg Object** (Resource)

6236 Configuration data that specifies how price responsive devices SHOULD respond to price
6237 changes while acting upon a given RateComponent.

6238 **consumeThreshold attribute** (Int32)

6239 Price responsive clients acting upon the associated RateComponent SHOULD consume the associated
6240 commodity while the price is less than this threshold.

6241 **maxReductionThreshold attribute** (Int32)

6242 Price responsive clients acting upon the associated RateComponent SHOULD reduce consumption to the
6243 maximum extent possible while the price is greater than this threshold.

6244 **PriceResponseCfgList Object** (List)

6245 A List element to hold PriceResponseCfg objects.

6246 **TimeConfiguration Object** ()

6247 Contains attributes related to the configuration of the time service.

6248 **dstEndRule attribute** (DstRuleType)

6249 Rule to calculate end of daylight savings time in the current year. Result of dstEndRule must be greater
6250 than result of dstStartRule.

6251 **dstOffset attribute** (TimeOffsetType)

6252 Daylight savings time offset from local standard time.

6253 **dstStartRule attribute** (DstRuleType)

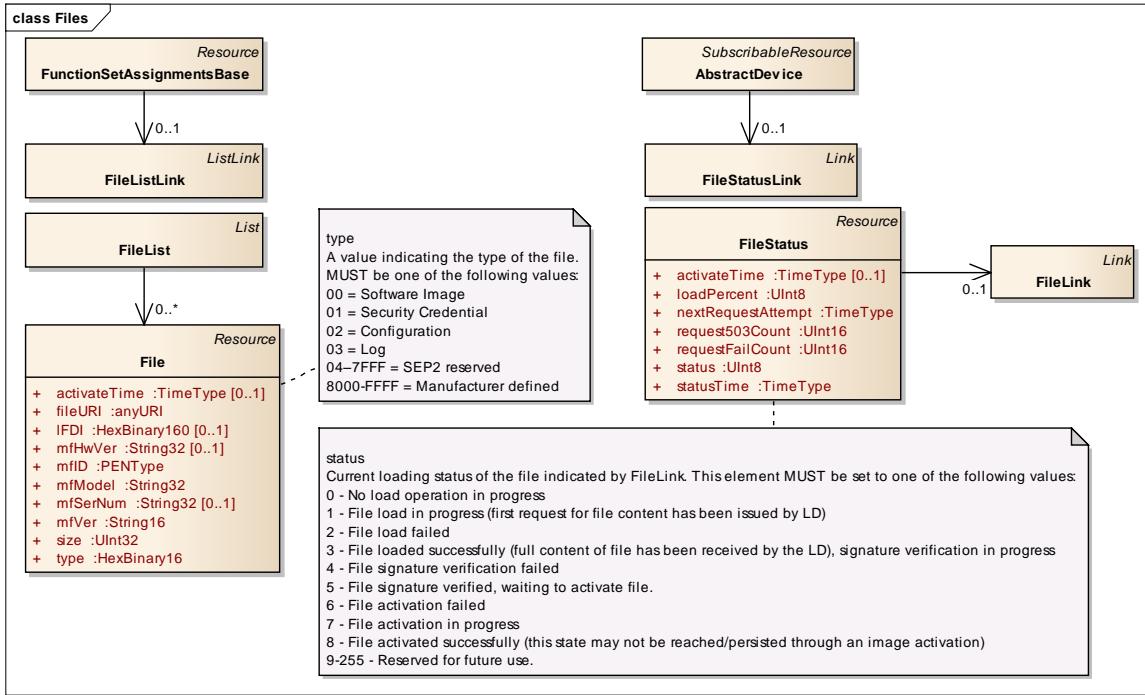
6254 Rule to calculate start of daylight savings time in the current year. Result of dstEndRule must be greater
6255 than result of dstStartRule.

6256 **tzOffset attribute** (TimeOffsetType)

6257 Local time zone offset from UTCTime. Does not include any daylight savings time offsets.

6258 15.1.13 SoftwareDownload Package

6259



6260

6261

Figure 15-20: Files**File Object (Resource)**

This resource contains various meta-data describing a file's characteristics. The meta-data provides general file information and also is used to support filtered queries of file lists

6265

activateTime attribute (TimeType) [0..1]

This element MUST be set to the date/time at which this file is activated. If the activation time is less than or equal to current time, the LD MUST immediately place the file into the activated state (in the case of a firmware file, the file is now the running image). If the activation time is greater than the current time, the LD MUST wait until the specified activation time is reached, then MUST place the file into the activated state. Omission of this element means that the LD MUST NOT take any action to activate the file until a subsequent GET to this File resource provides an activateTime.

6272

fileURI attribute (anyURI)

This element MUST be set to the URI location of the file binary artifact. This is the BLOB (binary large object) that is actually loaded by the LD

6275

IFDI attribute (HexBinary160) [0..1]

This element MUST be set to the LFDI of the device for which this file is targeted.

6277

mfHwVer attribute (String32) [0..1]

This element MUST be set to the hardware version for which this file is targeted.

6279

mfID attribute (PENType)

This element MUST be set to the manufacturer's Private Enterprise Number (assigned by IANA).

6281 *mfModel attribute* (*String32*)

6282 This element MUST be set to the manufacturer model number for which this file is targeted. The syntax
6283 and semantics are left to the manufacturer.

6284 *mfSerNum attribute* (*String32*) [0..1]

6285 This element MUST be set to the manufacturer serial number for which this file is targeted. The syntax
6286 and semantics are left to the manufacturer.

6287 *mfVer attribute* (*String16*)

6288 This element MUST be set to the software version information for this file. The syntax and semantics are
6289 left to the manufacturer.

6290 *size attribute* (*UInt32*)

6291 This element MUST be set to the total size (in bytes) of the file referenced by fileURI.

6292 *type attribute* (*HexBinary16*)

6293 A value indicating the type of the file. MUST be one of the following values:

6294 00 = Software Image

6295 01 = Security Credential

6296 02 = Configuration

6297 03 = Log

6298 04–7FFF = SEP2 reserved

6299 8000–FFFF = Manufacturer defined

6300 *FileList Object* (List)

6301 A List element to hold File objects.

6302 *FileStatus Object* (Resource)

6303 This object provides status of device file load and activation operations.

6304 *activateTime attribute* (*TimeType*) [0..1]

6305 Date/time at which this File, referred to by FileLink, will be activated. Omission of or presence and value
6306 of this element MUST exactly match omission or presence and value of the activateTime element from
6307 the File resource.

6308 *loadPercent attribute* (*UInt8*)

6309 This element MUST be set to the percentage of the file, indicated by FileLink, that was loaded during the
6310 latest load attempt. This value MUST be reset to 0 each time a load attempt is started for the File
6311 indicated by FileLink. This value MUST be increased when an LD receives HTTP response containing
6312 file content. This value MUST be set to 100 when the full content of the file has been received by the LD

6313 *nextRequestAttempt attribute* (*TimeType*)

6314 This element MUST be set to the time at which the LD will issue its next GET request for file content
6315 from the File indicated by FileLink

6316 *request503Count attribute* (*UInt16*)

6317 This value MUST be reset to 0 when FileLink is first pointed at a new File. This value MUST be
6318 incremented each time an

6319 LD receives a 503 error from the FS.

6320 ***requestFailCount attribute (UInt16)***

6321 This value MUST be reset to 0 when FileLink is first pointed at a new File. This value MUST be
6322 incremented each time a GET request for file content failed. 503 errors MUST be excluded from this
6323 counter.

6324 ***status attribute (UInt8)***

6325 Current loading status of the file indicated by FileLink. This element MUST be set to one of the following
6326 values:

6327 0 - No load operation in progress

6328 1 - File load in progress (first request for file content has been issued by LD)

6329 2 - File load failed

6330 3 - File loaded successfully (full content of file has been received by the LD), signature verification in
6331 progress

6332 4 - File signature verification failed

6333 5 - File signature verified, waiting to activate file.

6334 6 - File activation failed

6335 7 - File activation in progress

6336 8 - File activated successfully (this state may not be reached/persisted through an image activation)

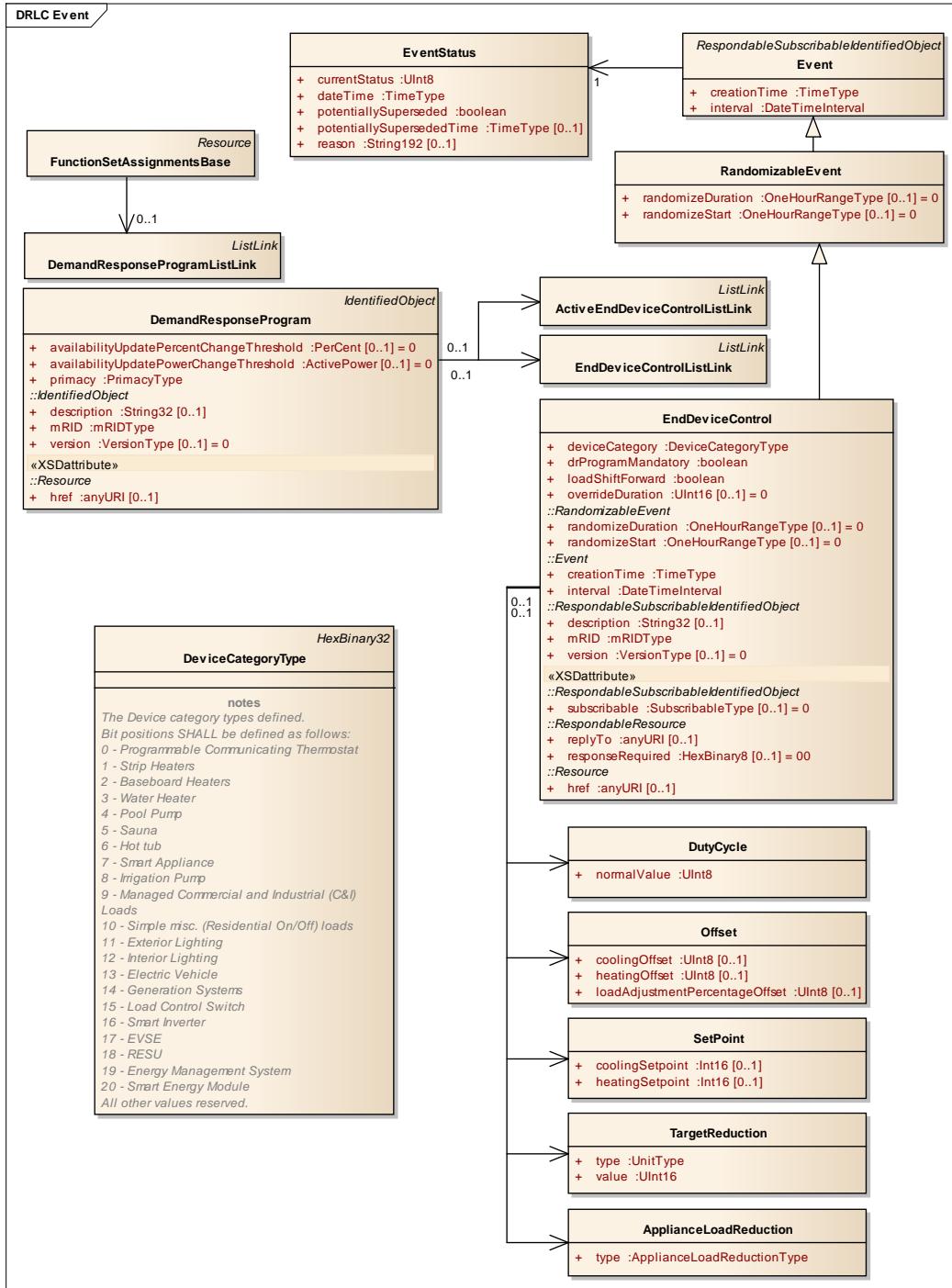
6337 9-255 - Reserved for future use.

6338 ***statusTime attribute (TimeType)***

6339 This element MUST be set to the time at which file status transitioned to the value indicated in the status
6340 element.

6341 15.1.14 **DRLC Package**

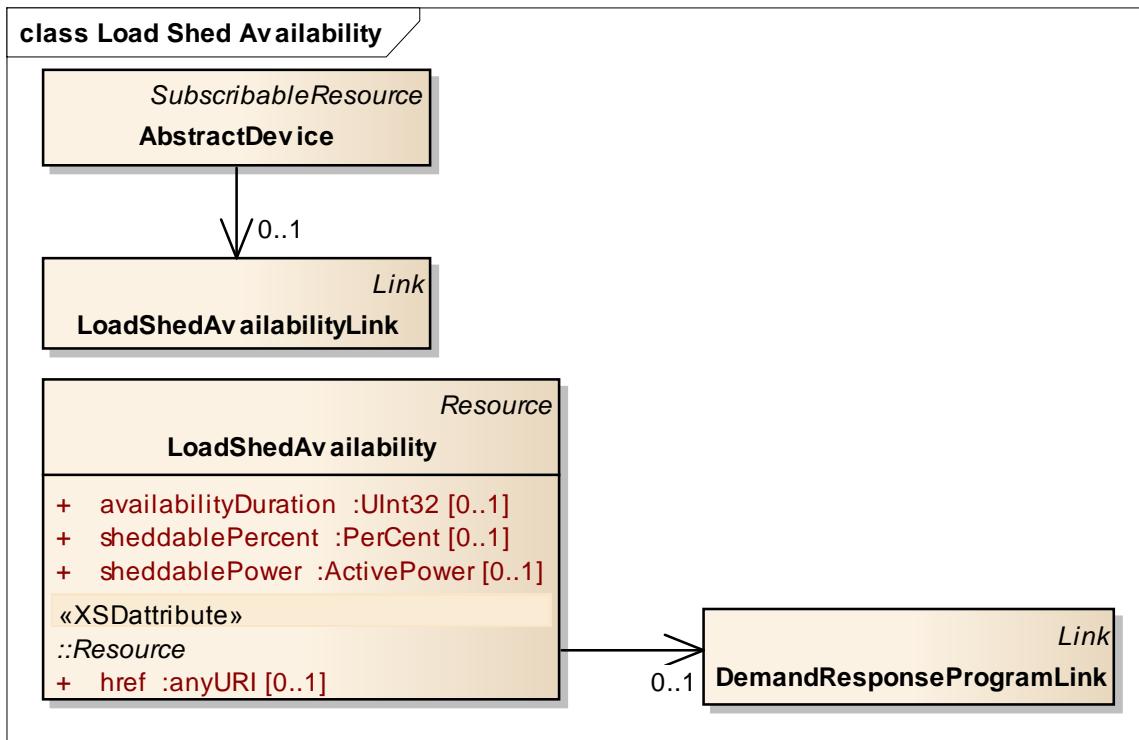
6342 Contains definitions for Demand Response Load Control functionality.



6343

6344

Figure 15-21: DRLC Event



6345

6346

Figure 15-22: Load Shed Availability**ApplianceLoadReduction Object ()**

The ApplianceLoadReduction object is used by a Demand Response service provider to provide signals for ENERGY STAR compliant appliances. See the definition of ApplianceLoadReductionType for more information.

6351

***type attribute* (ApplianceLoadReductionType)**

Indicates the type of appliance load reduction requested.

6353

DemandResponseProgram Object (IdentifiedObject)

Demand response program.

6355

***availabilityUpdatePercentChangeThreshold attribute* (PerCent) [0..1]**

This attribute allows program providers to specify the requested granularity of updates to LoadShedAvailability shippablePercent. If not present, or set to 0, then updates to LoadShedAvailability SHALL NOT be posted. If present and greater than zero, then clients SHALL post their LoadShedAvailability if it has not previously been posted, and thereafter if the difference between the previously posted value and the current value of LoadShedAvailability shippablePower is greater than availabilityUpdatePercentChangeThreshold.

6362

***availabilityUpdatePowerChangeThreshold attribute* (ActivePower) [0..1]**

This attribute allows program providers to specify the requested granularity of updates to LoadShedAvailability shippablePower. If not present, or set to 0, then updates to LoadShedAvailability SHALL NOT be posted. If present and greater than zero, then clients SHALL post their LoadShedAvailability if it has not previously been posted, and thereafter if the difference between the previously posted value and the current value of LoadShedAvailability shippablePower is greater than availabilityUpdatePowerChangeThreshold.

6369 *primacy attribute* (*PrimacyType*)

6370 Indicates the relative primacy of the provider of this program.

6371 *DemandResponseProgramList Object* (*SubscribableList*)

6372 A List element to hold DemandResponseProgram objects.

6373 *DutyCycle Object* ()

6374 Duty cycle control is a device specific issue and is managed by the device. The duty cycle of the
 6375 device under control should span the shortest practical time period in accordance with the nature
 6376 of the device under control and the intent of the request for demand reduction. The default
 6377 factory setting SHOULD be three minutes for each 10% of duty cycle. This indicates that the
 6378 default time period over which a duty cycle is applied is 30 minutes, meaning a 10% duty cycle
 6379 would cause a device to be ON for 3 minutes. The “off state” SHALL precede the “on state”.

6380 *normalValue attribute* (*UInt8*)

6381 Contains the maximum On state duty cycle applied by the end device, as a percentage of time. The field
 6382 not present indicates that this field has not been used by the end device.

6383 *EndDeviceControl Object* (*RandomizableEvent*)

6384 Instructs an EndDevice to perform a specified action.

6385 *deviceCategory attribute* (*DeviceCategoryType*)

6386 Specifies the bitmap indicating the categories of devices that SHOULD respond. Devices SHOULD
 6387 ignore events that do not indicate their device category.

6388 *drProgramMandatory attribute* (*boolean*)

6389 A flag to indicate if the EndDeviceControl is considered a mandatory event as defined by the service
 6390 provider issuing the EndDeviceControl. The drProgramMandatory flag alerts the client/user that they will
 6391 be subject to penalty or ineligibility based on the service provider’s program rules for that
 6392 EndDeviceCategory.

6393 *loadShiftForward attribute* (*boolean*)

6394 Indicates that the event intends to increase consumption. A value of true indicates the intention to increase
 6395 usage value, and a value of false indicates the intention to decrease usage.

6396 *overrideDuration attribute* (*UInt16*) [0..1]

6397 The overrideDuration attribute provides a duration, in seconds, for which a client device is allowed to
 6398 override this EndDeviceControl and still meet the contractual agreement with a service provider without
 6399 opting out. If overrideDuration is not specified, then it SHALL default to 0.

6400 *EndDeviceControlList Object* (*SubscribableList*)

6401 A List element to hold EndDeviceControl objects.

6402 *LoadShedAvailability Object* (*Resource*)

6403 Indicates current consumption status and ability to shed load.

6404 *availabilityDuration attribute* (*UInt32*) [0..1]

6405 Indicates for how many seconds the consuming device will be able to reduce consumption at the
 6406 maximum response level.

6407 *sheddablePercent attribute* (*PerCent*) [0..1]

6408 Maximum percent of current operating load that is estimated to be sheddable.

6409 *sheddablePower attribute* (*ActivePower*) [0..1]

6410 Maximum amount of current operating load that is estimated to be sheddable, in Watts.

Offset Object ()

6412 If a temperature offset is sent that causes the heating or cooling temperature set point to exceed
6413 the limit boundaries that are programmed into the device, the device SHALL respond by setting
6414 the temperature at the limit.

6415 If an EDC is being targeted at multiple devices or to a device that controls multiple devices (e.g.,
6416 EMS), it can provide multiple Offset types within one EDC. For events with multiple Offset
6417 types, a client SHALL select the Offset that best fits their operating function.

6418 Alternatively, an event with a single Offset type can be targeted at an EMS in order to request a
6419 percentage load reduction on the average energy usage of the entire premise. An EMS SHOULD
6420 use the Metering function set to determine the initial load in the premise, reduce energy
6421 consumption by controlling devices at its disposal, and at the conclusion of the event, once again
6422 use the Metering function set to determine if the desired load reduction was achieved.

coolingOffset attribute (UInt8) [0..1]

6423 The value change requested for the cooling offset, in degree C / 10. The value should be added to the
6424 normal set point for cooling, or if loadShiftForward is true, then the value should be subtracted from the
6425 normal set point.
6426

heatingOffset attribute (UInt8) [0..1]

6427 The value change requested for the heating offset, in degree C / 10. The value should be subtracted for
6428 heating, or if loadShiftForward is true, then the value should be added to the normal set point.
6429

loadAdjustmentPercentageOffset attribute (UInt8) [0..1]

6430 The value change requested for the load adjustment percentage, in tenths of a percent . The value should
6431 be subtracted from the normal setting, or if loadShiftForward is true, then the value should be added to the
6432 normal setting.
6433

SetPoint Object ()

6434 The SetPoint object is used to apply specific temperature set points to a temperature control
6435 device. The values of the heatingSetpoint and coolingSetpoint attributes SHALL be calculated as
6436 follows:
6437

6438 Cooling/Heating Temperature Set Point / 100 = temperature in degrees Celsius where -273.15°C
6439 <= temperature <= 327.67°C, corresponding to a Cooling and/or Heating Temperature Set Point.
6440 The maximum resolution this format allows is 0.01 °C.

6441 The field not present in a Response indicates that this field has not been used by the end device.

6442 If a temperature is sent that exceeds the temperature limit boundaries that are programmed into
6443 the device, the device SHALL respond by setting the temperature at the limit.

coolingSetpoint attribute (Int16) [0..1]

6444 This attribute represents the cooling temperature set point in degrees Celsius / 100. (Hundreds of a
6445 degree C)
6446

heatingSetpoint attribute (Int16) [0..1]

6447 This attribute represents the heating temperature set point in degrees Celsius / 100. (Hundreds of a
6448 degree C)
6449

6450 **TargetReduction Object ()**

6451 The TargetReduction object is used by a Demand Response service provider to provide a
 6452 RECOMMENDED threshold that a device/premises should maintain its consumption below. For
 6453 example, a service provider can provide a RECOMMENDED threshold of some kWh for a 3-
 6454 hour event. This means that the device/premises would maintain its consumption below the
 6455 specified limit for the specified period.

6456 **type attribute (UnitType)**

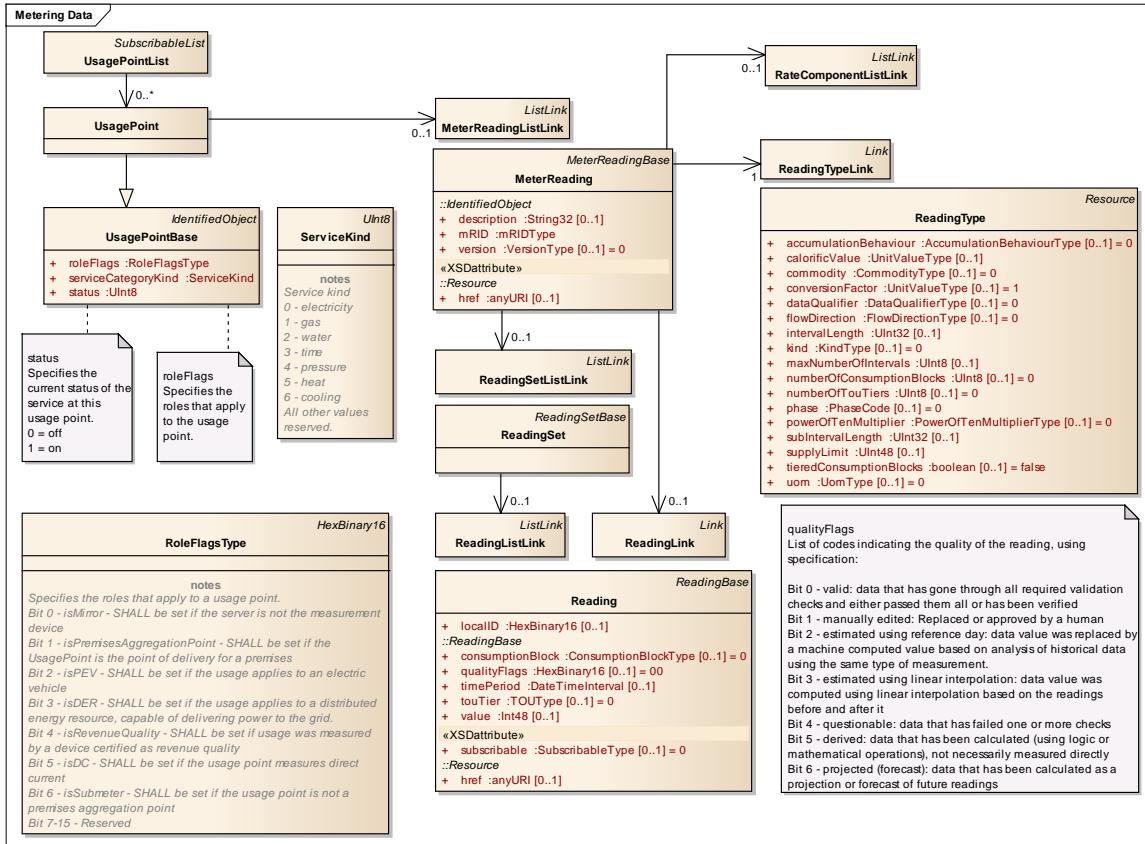
6457 Indicates the type of reduction requested.

6458 **value attribute (UInt16)**

6459 Indicates the requested amount of the relevant commodity to be reduced.

6460 15.1.15 **Metering Package**

6461 Contains definitions related to measurements of energy at usage points.



6462

6463

Figure 15-23: Metering Data

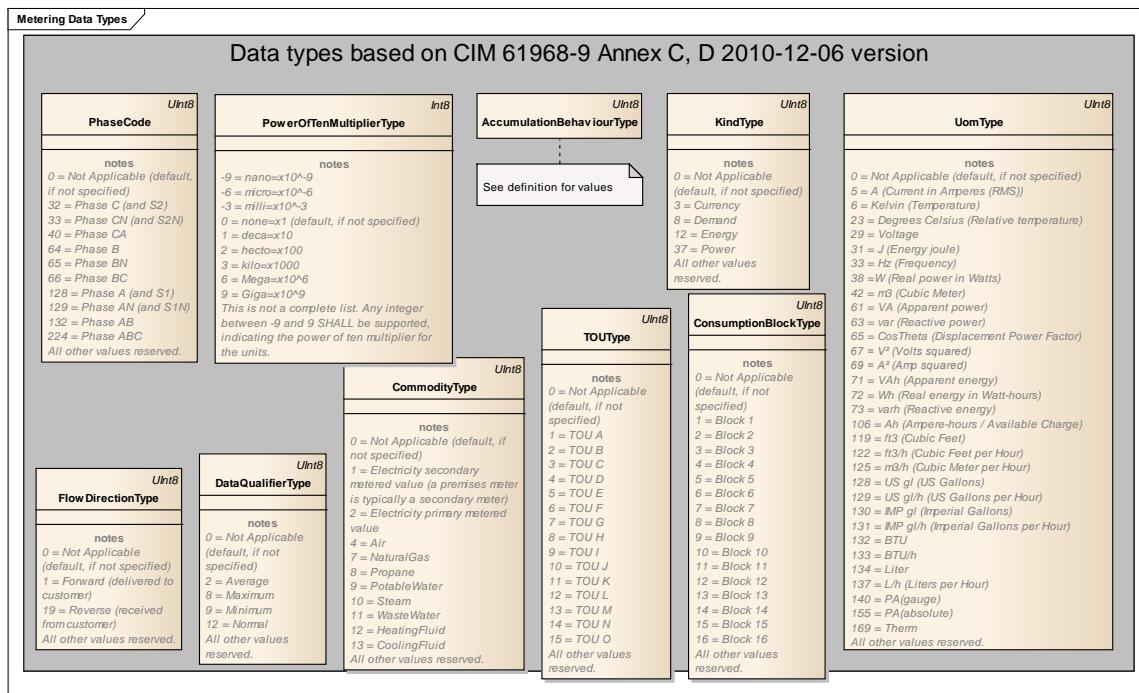


Figure 15-24: Metering Data Types

MeterReading Object (MeterReadingBase)

Set of values obtained from the meter.

MeterReadingList Object (SubscribableList)

A List element to hold MeterReading objects.

Reading Object (ReadingBase)

Specific value measured by a meter or other asset.

localID attribute (HexBinary16) [0..1]

The local identifier for this reading within the reading set. localIDs are assigned in order of creation time. For interval data, this value SHALL increase with each interval time, and for block/tier readings, localID SHALL not be specified.

subscribable attribute (SubscribableType) [0..1] «XSDataattribute»

Indicates whether or not subscriptions are supported for this resource, and whether or not conditional (thresholds) are supported. If not specified, is "not subscribable" (0).

ReadingList Object (SubscribableList)

A List element to hold Reading objects.

ReadingSet Object (ReadingSetBase)

A set of Readings of the ReadingType indicated by the parent MeterReading.

ReadingSetList Object (SubscribableList)

A List element to hold ReadingSet objects.

ReadingType Object (Resource)

Type of data conveyed by a specific Reading. See IEC 61968 Part 9 Annex C for full definitions

6487 of these values.

6488 *accumulationBehaviour attribute* (*AccumulationBehaviourType*) [0..1]

6489 The “accumulation behaviour” indicates how the value is represented to accumulate over time.

6490 *calorificValue attribute* (*UnitValueType*) [0..1]

6491 The amount of heat generated when a given mass of fuel is completely burned. The CalorificValue is used
6492 to convert the measured volume or mass of gas into kWh. The CalorificValue attribute represents the
6493 current active value.

6494 *commodity attribute* (*CommodityType*) [0..1]

6495 Indicates the commodity applicable to this ReadingType.

6496 *conversionFactor attribute* (*UnitValueType*) [0..1]

6497 Accounts for changes in the volume of gas based on temperature and pressure. The ConversionFactor
6498 attribute represents the current active value. The ConversionFactor is dimensionless. The default value for
6499 the ConversionFactor is 1, which means no conversion is applied. A price server can advertise a
6500 new/different value at any time.

6501 *dataQualifier attribute* (*DataQualifierType*) [0..1]

6502 The data type can be used to describe a salient attribute of the data. Possible values are average, absolute,
6503 and etc.

6504 *flowDirection attribute* (*FlowDirectionType*) [0..1]

6505 Anything involving current might have a flow direction. Possible values include forward and reverse.

6506 *intervalLength attribute* (*UInt32*) [0..1]

6507 Default interval length specified in seconds.

6508 *kind attribute* (*KindType*) [0..1]

6509 Compound class that contains kindCategory and kindIndex

6510 *maxNumberOfIntervals attribute* (*UInt8*) [0..1]

6511 To be populated for mirrors of interval data to set the expected number of intervals per ReadingSet.
6512 Servers may discard intervals received that exceed this number.

6513 *numberOfConsumptionBlocks attribute* (*UInt8*) [0..1]

6514 Number of consumption blocks. 0 means not applicable, and is the default if not specified. The value
6515 needs to be at least 1 if any actual prices are provided.

6516 *numberOfTouTiers attribute* (*UInt8*) [0..1]

6517 The number of TOU tiers that can be used by any resource configured by this ReadingType. Servers
6518 SHALL populate this value with the largest touTier value that will ever be used while this ReadingType
6519 is in effect. Servers SHALL set numberOfTouTiers equal to the number of standard TOU tiers plus the
6520 number of CPP tiers that may be used while this ReadingType is in effect. Servers SHALL specify a
6521 value between 1 and 255 (inclusive) for numberOfTouTiers (servers providing flat rate pricing SHALL
6522 set numberOfTouTiers to 1, as in practice there is no difference between having no tiers and having one
6523 tier).

6524 *phase attribute* (*PhaseCode*) [0..1]

6525 Contains phase information associated with the type.

6526 *powerOfTenMultiplier attribute* (*PowerOfTenMultiplierType*) [0..1]

6527 Indicates the power of ten multiplier applicable to the unit of measure of this ReadingType.

6528 *subIntervalLength attribute (UInt32) [0..1]*

6529 Default sub-interval length specified in seconds for Readings of ReadingType. Some demand calculations
6530 are done over a number of smaller intervals. For example, in a rolling demand calculation, the demand
6531 value is defined as the rolling sum of smaller intervals over the intervalLength. The subintervalLength is
6532 the length of the smaller interval in this calculation. It SHALL be an integral division of the
6533 intervalLength. The number of sub-intervals can be calculated by dividing the intervalLength by the
6534 subintervalLength.

6535 *supplyLimit attribute (UInt48) [0..1]*

6536 Reflects the supply limit set in the meter. This value can be compared to the Reading value to understand
6537 if limits are being approached or exceeded. Units follow the same definition as in this ReadingType.

6538 *tieredConsumptionBlocks attribute (boolean) [0..1]*

6539 Specifies whether or not the consumption blocks are differentiated by TOUTier or not. Default is false, if
6540 not specified.

6541 true = consumption accumulated over individual tiers

6542 false = consumption accumulated over all tiers

6543 *uom attribute (UomType) [0..1]*

6544 Indicates the measurement type for the units of measure for the readings of this type.

6545 *UsagePoint Object (UsagePointBase)*

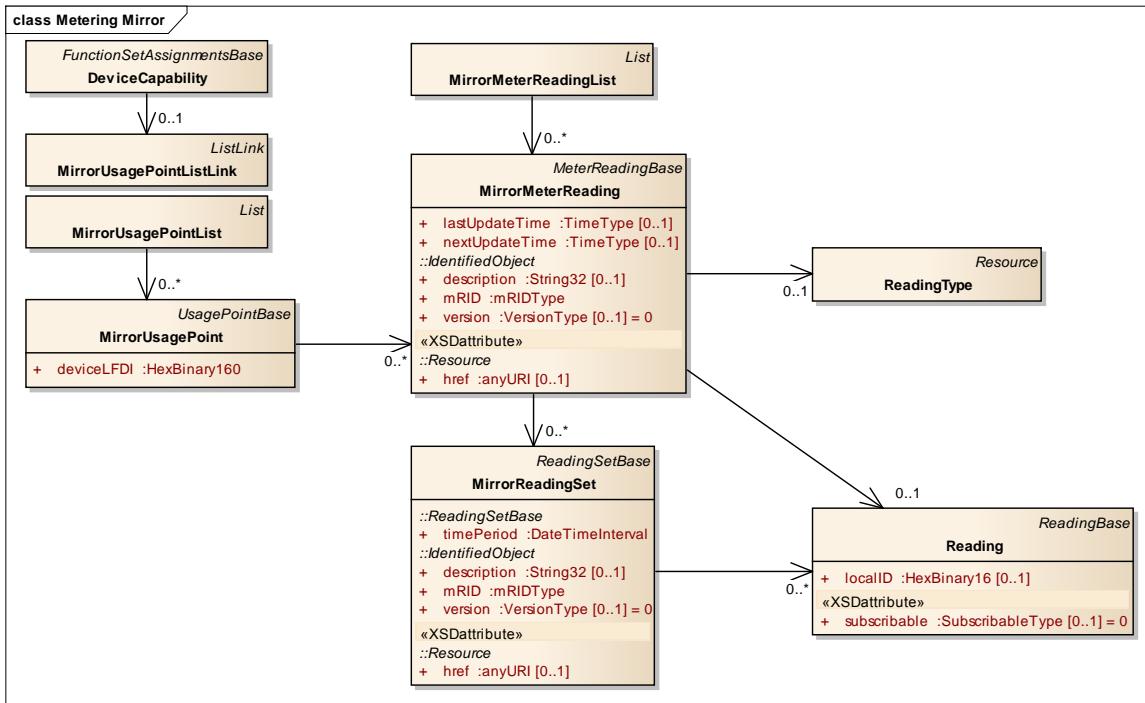
6546 Logical point on a network at which consumption or production is either physically measured
6547 (e.g. metered) or estimated (e.g. unmetered street lights).

6548 *UsagePointList Object (SubscribableList)*

6549 A List element to hold UsagePoint objects.

6550 15.1.15.1 Metering Mirror Package

6551



6552

6553

Figure 15-25: Metering Mirror

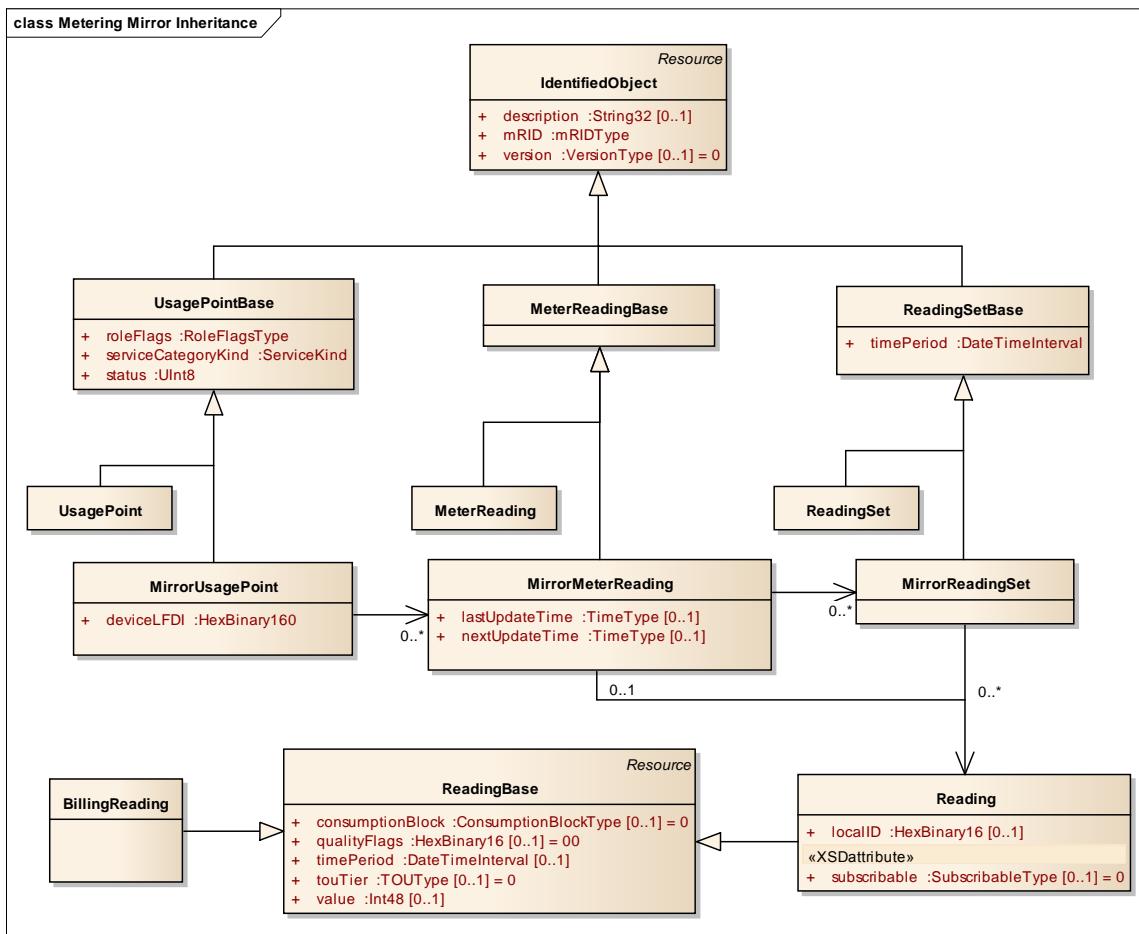


Figure 15-26: Metering Mirror Inheritance

MirrorMeterReading Object (MeterReadingBase)

6556 Mimic of MeterReading used for managing mirrors.

lastUpdateTime attribute (TimeType) [0..1]

6558 The date and time of the last update.

nextUpdateTime attribute (TimeType) [0..1]

6560 The date and time of the next planned update.

MirrorMeterReadingList Object (List)

6563 A List of MirrorMeterReading instances.

MeterReadingBase Object (IdentifiedObject)

6565 A container for associating ReadingType, Readings and ReadingSets.

MirrorReadingSet Object (ReadingSetBase)

6567 A set of Readings of the ReadingType indicated by the parent MeterReading.

MirrorUsagePoint Object (UsagePointBase)

6568 A parallel to UsagePoint to support mirroring

deviceLFDI attribute (HexBinary160)

6571 The LFDI of the device being mirrored.

MirrorUsagePointList Object (List)

6573 A List of MirrorUsagePoint instances.

ReadingBase Object (Resource)

6575 Specific value measured by a meter or other asset. ReadingBase is abstract, used to define the
6576 elements common to Reading and IntervalReading.

consumptionBlock attribute (*ConsumptionBlockType*) [0..1]

6578 Indicates the consumption block related to the reading. REQUIRED if ReadingType
6579 numberOfConsumptionBlocks is non-zero. If not specified, is assumed to be "0 - N/A".

qualityFlags attribute (*HexBinary16*) [0..1]

6580 List of codes indicating the quality of the reading, using specification:

6582 Bit 0 - valid: data that has gone through all required validation checks and either passed them all or has
6583 been verified

6584 Bit 1 - manually edited: Replaced or approved by a human

6585 Bit 2 - estimated using reference day: data value was replaced by a machine computed value based on
6586 analysis of historical data using the same type of measurement.

6587 Bit 3 - estimated using linear interpolation: data value was computed using linear interpolation based on
6588 the readings before and after it

6589 Bit 4 - questionable: data that has failed one or more checks

6590 Bit 5 - derived: data that has been calculated (using logic or mathematical operations), not necessarily
6591 measured directly

6592 Bit 6 - projected (forecast): data that has been calculated as a projection or forecast of future readings

timePeriod attribute (*DateTimeInterval*) [0..1]

6593 The time interval associated with the reading. If not specified, then defaults to the intervalLength
6594 specified in the associated ReadingType.

touTier attribute (*TOUType*) [0..1]

6595 Indicates the time of use tier related to the reading. REQUIRED if ReadingType numberOfTouTiers is
6596 non-zero. If not specified, is assumed to be "0 - N/A".

value attribute (*Int48*) [0..1]

6597 Value in units specified by ReadingType

ReadingSetBase Object (IdentifiedObject)

6602 A set of Readings of the ReadingType indicated by the parent MeterReading. ReadingBase is
6603 abstract, used to define the elements common to ReadingSet and IntervalBlock.

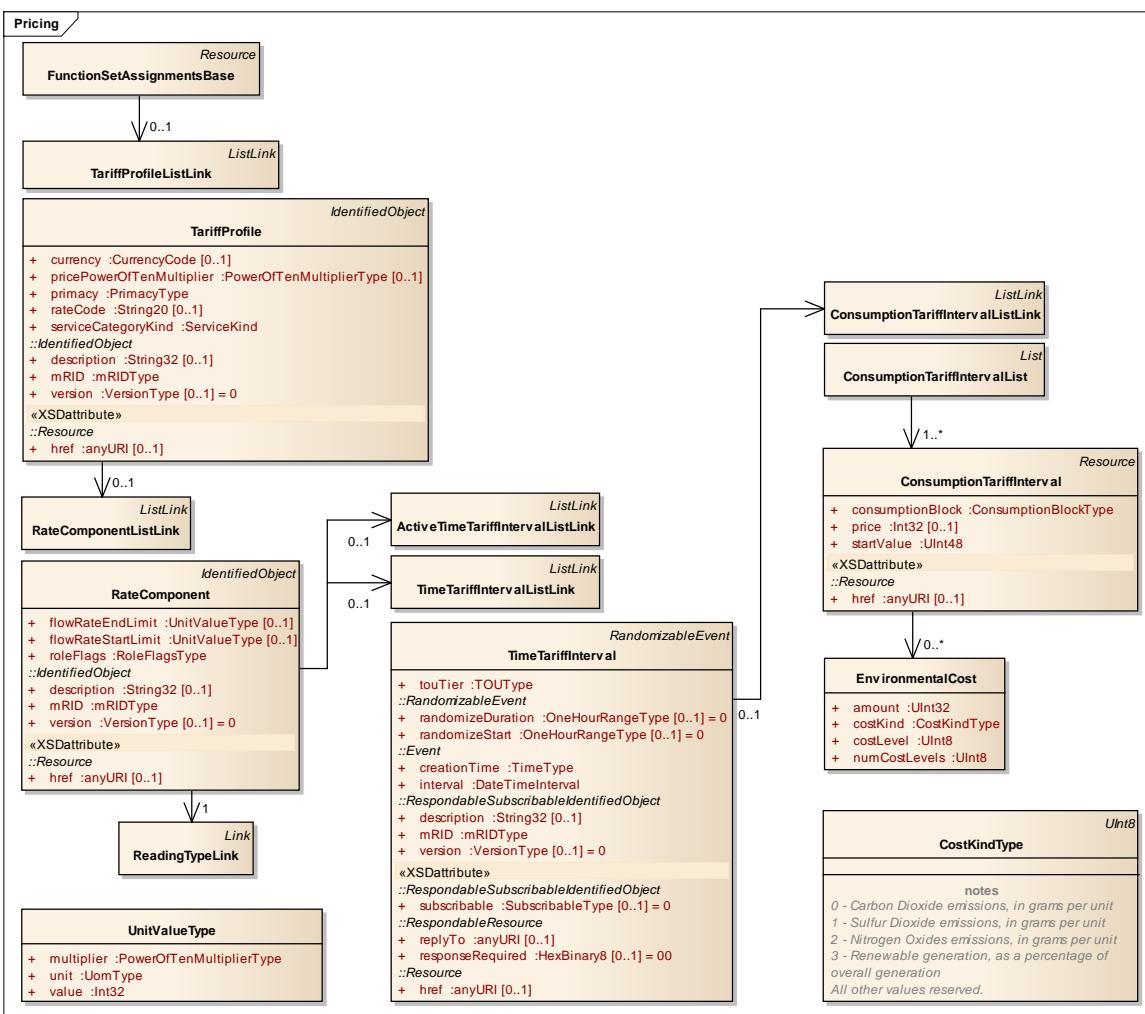
timePeriod attribute (*DateTimeInterval*)

6604 Specifies the time range during which the contained readings were taken.

UsagePointBase Object (IdentifiedObject)

6605 Logical point on a network at which consumption or production is either physically measured
(e.g. metered) or estimated (e.g. unmetered street lights). A container for associating
6606 ReadingType, Readings and ReadingSets.

- 6610 ***roleFlags attribute (RoleFlagsType)***
 6611 Specifies the roles that apply to the usage point.
- 6612 ***serviceCategoryKind attribute (ServiceKind)***
 6613 The kind of service provided by this usage point.
- 6614 ***status attribute (UInt8)***
 6615 Specifies the current status of the service at this usage point.
 0 = off
 1 = on
- 6618 **15.1.16 Pricing Package**
 6619 Contains definitions of information related to price.

**Figure 15-27: Pricing**

- 6620
 6621
ConsumptionTariffInterval Object (Resource)
 6622 One of a sequence of thresholds defined in terms of consumption quantity of a service such as electricity, water, gas, etc. It defines the steps or blocks in a step tariff structure, where startValue

6625 simultaneously defines the entry value of this step and the closing value of the previous step.
 6626 Where consumption is greater than startValue, it falls within this block and where consumption is
 6627 less than or equal to startValue, it falls within one of the previous blocks.

consumptionBlock attribute (ConsumptionBlockType)

6628 Indicates the consumption block related to the reading. If not specified, is assumed to be "0 - N/A".

price attribute (Int32) [0..1]

6631 The charge for this rate component, per unit of measure defined by the associated ReadingType, in
 6632 currency specified in TariffProfile.

6633 The Pricing service provider determines the appropriate price attribute value based on its applicable
 6634 regulatory rules. For example, price could be net or inclusive of applicable taxes, fees, or levies.

6635 The Billing function set provides the ability to represent billing information in a more detailed manner.

startValue attribute (UInt48)

6637 The lowest level of consumption that defines the starting point of this consumption step or block.
 6638 Thresholds start at zero for each billing period.

6639 If specified, the first ConsumptionTariffInterval.startValue for a TimeTariffInterval instance SHALL begin
 6640 at "0." Subsequent ConsumptionTariffInterval.startValue elements SHALL be greater than the previous
 6641 one.

ConsumptionTariffIntervalList Object (List)

6642 A List element to hold ConsumptionTariffInterval objects.

CostKindType Object (UInt8)

6643 0 - Carbon Dioxide emissions, in grams per unit

6646 1 - Sulfur Dioxide emissions, in grams per unit

6647 2 - Nitrogen Oxides emissions, in grams per unit

6648 3 - Renewable generation, as a percentage of overall generation

6649 All other values reserved.

EnvironmentalCost Object ()

6651 Provides alternative or secondary price information for the relevant RateComponent. Supports
 6652 jurisdictions that seek to convey the environmental price per unit of the specified commodity not
 6653 expressed in currency.

6654 Implementers and consumers can use this attribute to prioritize operations of their HAN devices
 6655 (e.g., PEV charging during times of high availability of renewable electricity resources).

amount attribute (UInt32)

6657 The estimated or actual environmental or other cost, per commodity unit defined by the ReadingType, for
 6658 this RateComponent (e.g., grams of carbon dioxide emissions each per kWh).

costKind attribute (CostKindType)

6660 The kind of cost referred to in the amount.

costLevel attribute (UInt8)

6662 The relative level of the amount attribute. In conjunction with numCostLevels, this attribute informs a
 6663 device of the relative scarcity of the amount attribute (e.g., a high or low availability of renewable
 6664 generation).

6665 numCostLevels and costLevel values SHALL ascend in order of scarcity, where "0" signals the lowest
 6666 relative cost and higher values signal increasing cost. For example, if numCostLevels is equal to "3,"
 6667 then if the lowest relative costLevel were equal to "0," devices would assume this is the lowest relative
 6668 period to operate. Likewise, if the costLevel in the next TimeTariffInterval instance is equal to "1," then
 6669 the device would assume it is relatively more expensive, in environmental terms, to operate during this
 6670 TimeTariffInterval instance than the previous one.

6671 There is no limit to the number of relative price levels other than that indicated in the attribute type, but
 6672 for practicality, service providers should strive for simplicity and recognize the diminishing returns
 6673 derived from increasing the numCostLevel value greater than four.

6674 *numCostLevels attribute (UInt8)*

6675 The number of all relative cost levels.

6676 In conjunction with costLevel, numCostLevels signals the relative scarcity of the commodity for the
 6677 duration of the TimeTariffInterval instance (e.g., a relative indication of cost). This is useful in providing
 6678 context for nominal cost signals to consumers or devices that might see a range of amount values from
 6679 different service providers or from the same service provider.

6680 *RateComponent Object (IdentifiedObject)*

6681 Specifies the applicable charges for a single component of the rate, which could be generation
 6682 price or consumption price, for example.

6683 *flowRateEndLimit attribute (UnitValueType) [0..1]*

6684 Specifies the maximum flow rate (e.g. kW for electricity) for which this RateComponent applies, for the
 6685 usage point and given rate / tariff.

6686 In combination with flowRateStartLimit, allows a service provider to define the demand or output
 6687 characteristics for the particular tariff design. If a server includes the flowRateEndLimit attribute, then it
 6688 SHALL also include flowRateStartLimit attribute.

6689 For example, a service provider's tariff limits customers to 20 kWs of demand for the given rate structure.
 6690 Above this threshold (from 20-50 kWs), there are different demand charges per unit of consumption. The
 6691 service provider can use flowRateStartLimit and flowRateEndLimit to describe the demand
 6692 characteristics of the different rates. Similarly, these attributes can be used to describe limits on premises
 6693 DERs that might be producing a commodity and sending it back into the distribution network.

6694 Note: At the time of writing, service provider tariffs with demand-based components were not originally
 6695 identified as being in scope, and service provider tariffs vary widely in their use of demand components
 6696 and the method for computing charges. It is expected that industry groups (e.g., OpenSG) will document
 6697 requirements in the future that the SEP 2.0 community can then use as source material for the next
 6698 version of SEP 2.0.

6699 *flowRateStartLimit attribute (UnitValueType) [0..1]*

6700 Specifies the minimum flow rate (e.g., kW for electricity) for which this RateComponent applies, for the
 6701 usage point and given rate / tariff.

6702 In combination with flowRateEndLimit, allows a service provider to define the demand or output
 6703 characteristics for the particular tariff design. If a server includes the flowRateStartLimit attribute, then it

6704 SHALL also include flowRateEndLimit attribute.

6705 ***roleFlags attribute (RoleFlagsType)***

6706 Specifies the roles that this usage point has been assigned.

6707 **TariffProfile Object (List)**

6708 A List element to hold TariffProfile objects.

6709 **TariffProfile Object (IdentifiedObject)**

6710 A schedule of charges; structure that allows the definition of tariff structures such as step (block)
6711 and time of use (tier) when used in conjunction with TimeTariffInterval and
6712 ConsumptionTariffInterval.

6713 ***currency attribute (CurrencyCode) [0..1]***

6714 The currency code indicating the currency for this TariffProfile.

6715 ***pricePowerOfTenMultiplier attribute (PowerOfTenMultiplierType) [0..1]***

6716 Indicates the power of ten multiplier for the price attribute.

6717 ***primacy attribute (PrimacyType)***

6718 Indicates the relative primacy of the provider of this program.

6719 ***rateCode attribute (String20) [0..1]***

6720 The rate code for this tariff profile. Provided by the Pricing service provider per its internal business
6721 needs and practices and provides a method to identify the specific rate code for the TariffProfile instance.
6722 This would typically not be communicated to the user except to facilitate troubleshooting due to its
6723 service provider-specific technical nature.

6724 ***serviceCategoryKind attribute (ServiceKind)***

6725 The kind of service provided by this usage point.

6726 **TariffProfileList Object (SubscribableList)**

6727 A List element to hold TariffProfile objects.

6728 **TimeTariffInterval Object (RandomizableEvent)**

6729 Describes the time-differentiated portion of the RateComponent, if applicable, and provides the
6730 ability to specify multiple time intervals, each with its own consumption-based components and
6731 other attributes.

6732 ***touTier attribute (TOUType)***

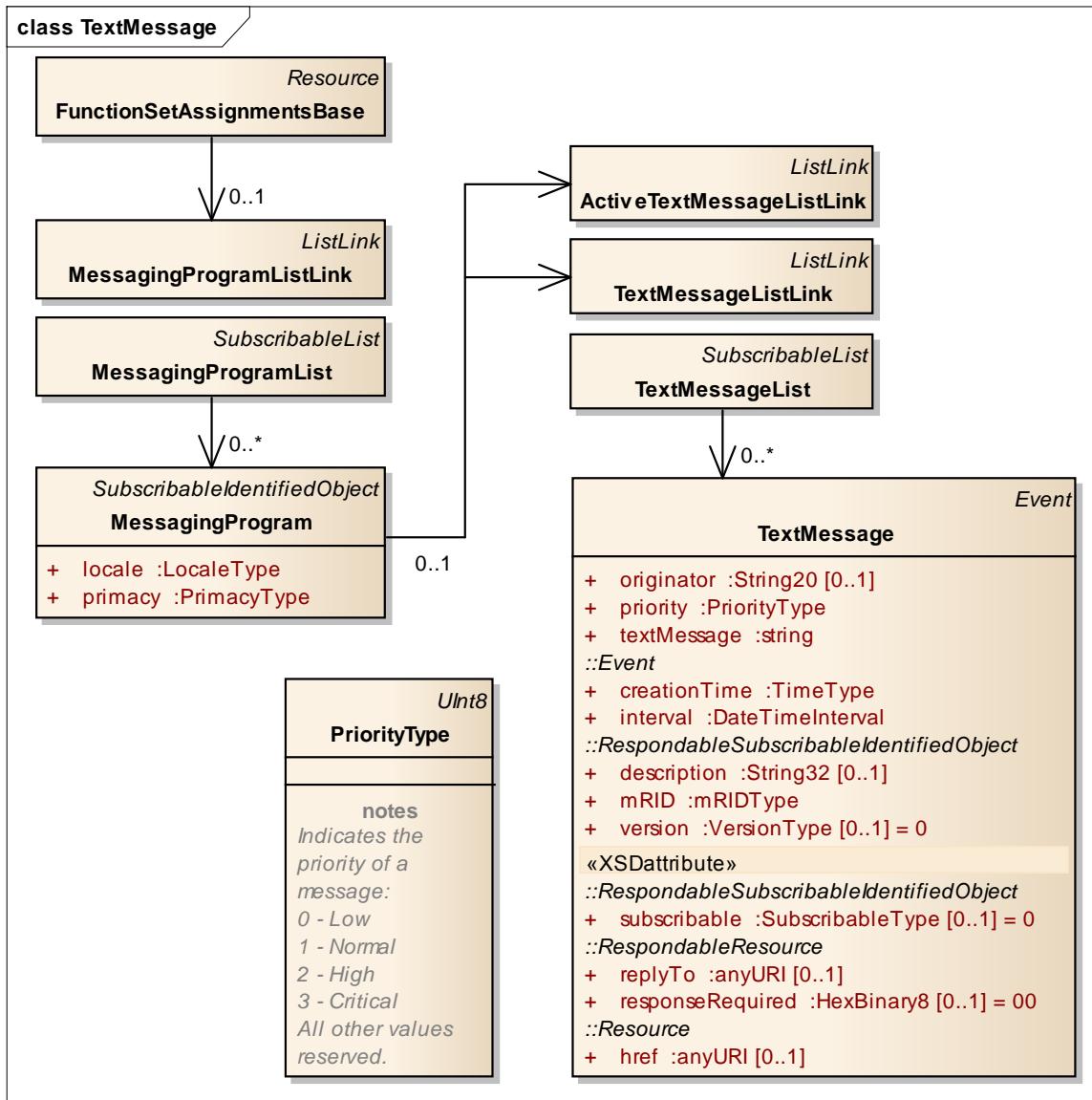
6733 Indicates the time of use tier related to the reading. If not specified, is assumed to be "0 - N/A".

6734 **TimeTariffIntervalList Object (SubscribableList)**

6735 A List element to hold TimeTariffInterval objects.

6736 15.1.17 **Messaging Package**

6737 Contains text message definitions.



6738

Figure 15-28: TextMessage6739 **MessagingProgram Object** (SubscribableIdentifiedObject)

6740 Provides a container for collections of text messages.

6741 **locale attribute** (**LocaleType**)

6742 Indicates the language and region of the messages in this collection.

6743 **primacy attribute** (**PrimacyType**)

6744 Indicates the relative primacy of the provider of this program.

6746 **MessagingProgramList Object** (SubscribableList)

6747 A List element to hold MessagingProgram objects.

6748 **PriorityType Object** (UInt8)

6749 Indicates the priority of a message:

6750 0 - Low

6751 1 - Normal

6752 2 - High

6753 3 - Critical

6754 All other values reserved.

6755 **TextMessage Object** (Event)

6756 Text message such as a notification.

6757 **originator attribute** (String20) [0..1]

6758 Indicates the human-readable name of the publisher of the message

6759 **priority attribute** (PriorityType)

6760 The priority is used to inform the client of the priority of the particular message. Devices with
6761 constrained or limited resources for displaying Messages should use this attribute to determine how to
6762 handle displaying currently active Messages (e.g. if a device uses a scrolling method with a single
6763 Message viewable at a time it MAY want to push a low priority Message to the background and bring a
6764 newly received higher priority Message to the foreground).

6765 **textMessage attribute** (string)

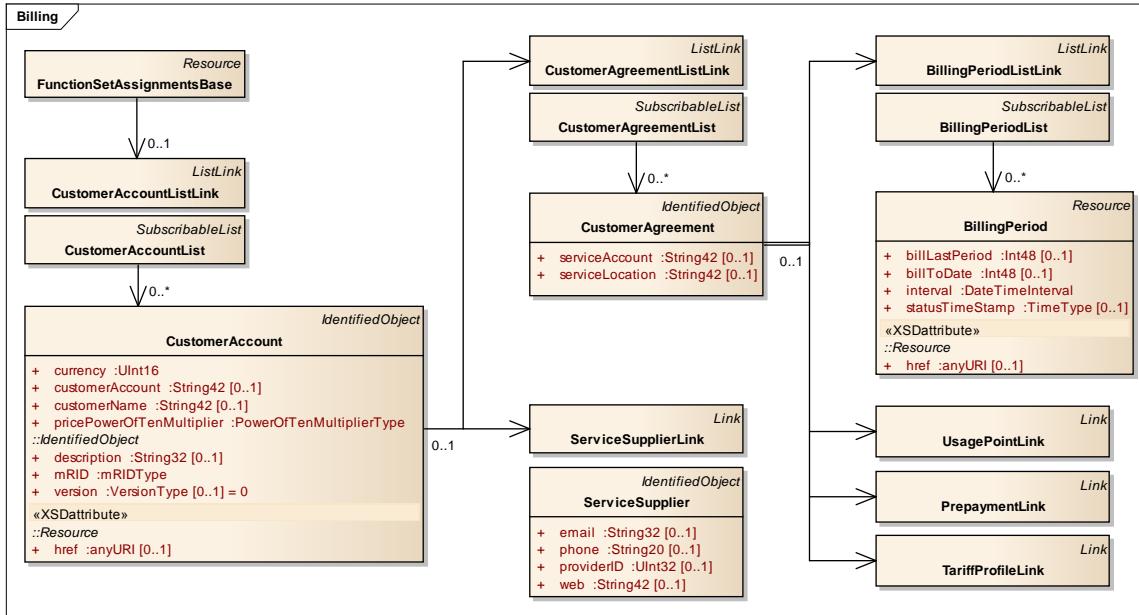
6766 The textMessage attribute contains the actual UTF-8 encoded text to be displayed in conjunction with the
6767 messageLength attribute which contains the overall length of the textMessage attribute. Clients and
6768 servers SHALL support a reception of a Message of 100 bytes in length. Messages that exceed the clients
6769 display size will be left to the client to choose what method to handle the message (truncation, scrolling,
6770 etc.).

6771 **TextMessageList Object** (SubscribableList)

6772 A List element to hold TextMessage objects.

6773 15.1.18 **Billing Package**

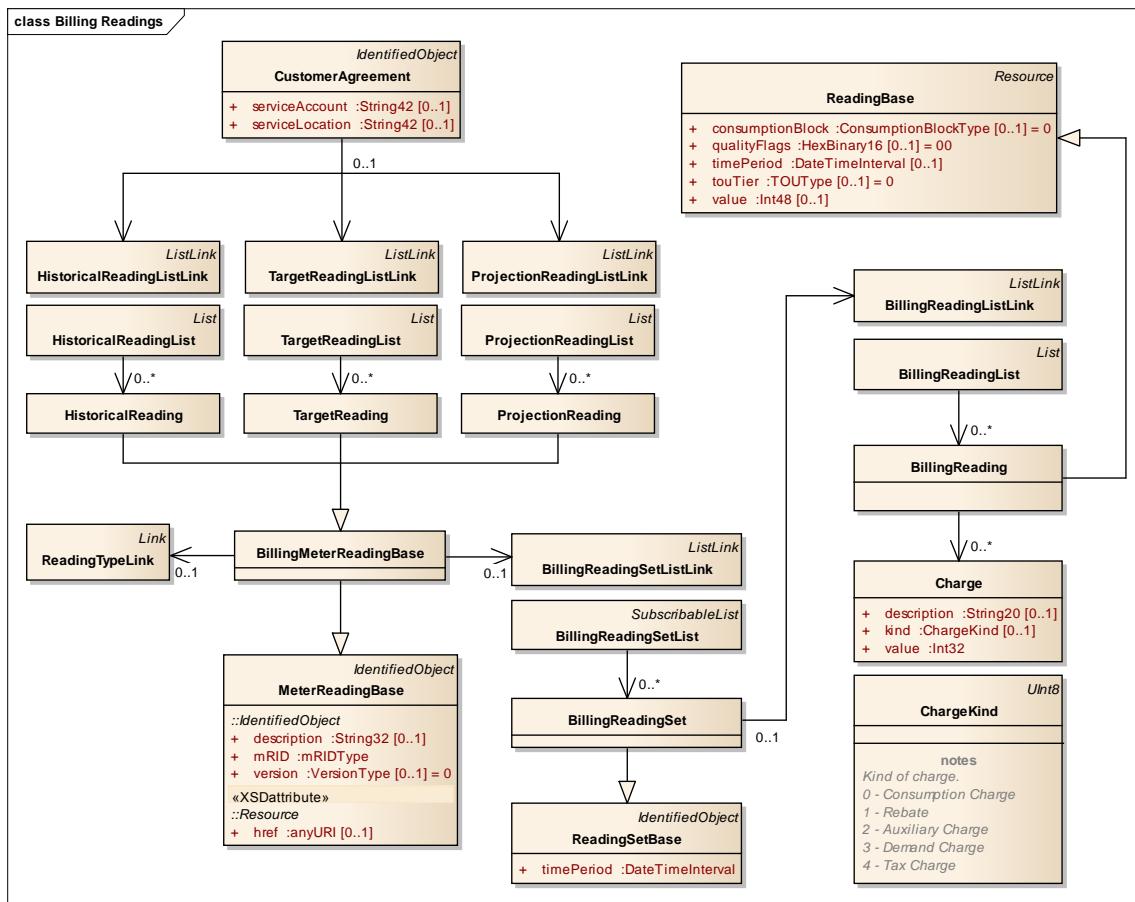
6774 Contains representations of charges and other billing related information.



6775

6776

Figure 15-29: Billing



6777

6778

Figure 15-30: Billing Readings**BillingPeriod Object** (Resource)

A Billing Period relates to the period of time on which a customer is billed. As an example the billing period interval for a particular customer might be 31 days starting on July 1, 2011. The start date and interval can change on each billing period. There may also be multiple billing periods related to a customer agreement to support different tariff structures.

6784 ***billLastPeriod attribute*** (*Int48*) [0..1]

The amount of the bill for the previous billing period.

6786 ***billToDate attribute*** (*Int48*) [0..1]

The bill amount related to the billing period as of the statusTimeStamp.

6788 ***interval attribute*** (*DateTimeInterval*)

The time interval for this billing period.

6790 ***statusTimeStamp attribute*** (*TimeType*) [0..1]

The date / time of the last update of this resource.

BillingPeriodList Object (SubscribableList)

A List element to hold BillingPeriod objects.

6794 BillingMeterReadingBase Object (MeterReadingBase)

6795 Contains historical, target, and projection readings of various types, possibly associated with
6796 charges.

6797 BillingReading Object (ReadingBase)

6798 Data captured at regular intervals of time. Interval data could be captured as incremental data,
6799 absolute data, or relative data. The source for the data is usually a tariff quantity or an engineering
6800 quantity. Data is typically captured in time-tagged, uniform, fixed-length intervals of 5 min, 10
6801 min, 15 min, 30 min, or 60 min. However, consumption aggregations can also be represented
6802 with this class.

6803 BillingReadingList Object (List)

6804 A List element to hold BillingReading objects.

6805 BillingReadingSet Object (ReadingSetBase)

6806 Time sequence of readings of the same reading type.

6807 BillingReadingSetList Object (SubscribableList)

6808 A List element to hold BillingReadingSet objects.

6809 Charge Object ()

6810 Charges contain charges on a customer bill. These could be items like taxes, levies, surcharges,
6811 rebates, or others. This is meant to allow the HAN device to retrieve enough information to be
6812 able to reconstruct an estimate of what the total bill would look like.

6813 Providers can provide line item billing, including multiple charge kinds (e.g. taxes, surcharges) at
6814 whatever granularity desired, using as many Charges as desired during a billing period. There can
6815 also be any number of Charges associated with different ReadingTypes to distinguish between
6816 TOU tiers, consumption blocks, or demand charges.

6817 description attribute (String20) [0..1]

6818 A description of the charge.

6819 kind attribute (ChargeKind) [0..1]

6820 The type (kind) of charge.

6821 value attribute (Int32)

6822 A monetary charge.

6823 ChargeKind Object (UInt8)

6824 Kind of charge.

6825 0 - Consumption Charge

6826 1 - Rebate

6827 2 - Auxiliary Charge

6828 3 - Demand Charge

6829 4 - Tax Charge

6830 CustomerAccount Object (IdentifiedObject)

6831 Assignment of a group of products and services purchased by the Customer through a
6832 CustomerAgreement, used as a mechanism for customer billing and payment. It contains common
6833 information from the various types of CustomerAgreements to create billings (invoices) for a

6834 Customer and receive payment.

currency attribute (*UInt16*)

6836 The ISO 4217 code indicating the currency applicable to the bill amounts in the summary. See list at
 6837 http://www.unece.org/cefact/recommendations/rec09/rec09_ecetrd203.pdf

customerAccount attribute (*String42*) [0..1]

6839 The account number for the customer (if applicable).

customerName attribute (*String42*) [0..1]

6841 The name of the customer.

pricePowerOfTenMultiplier attribute (*PowerOfTenMultiplierType*)

6842 Indicates the power of ten multiplier for the prices in this function set.

CustomerAccountList Object (*SubscribableList*)

6844 A List element to hold CustomerAccount objects.

CustomerAgreement Object (*IdentifiedObject*)

6845 Agreement between the customer and the service supplier to pay for service at a specific service location. It records certain billing information about the type of service provided at the service location and is used during charge creation to determine the type of service.

serviceAccount attribute (*String42*) [0..1]

6850 The account number of the service account (if applicable).

serviceLocation attribute (*String42*) [0..1]

6853 The address or textual description of the service location.

CustomerAgreementList Object (*SubscribableList*)

6854 A List element to hold CustomerAgreement objects.

HistoricalReading Object (*BillingMeterReadingBase*)

6855 To be used to present readings that have been processed and possibly corrected (as allowed, due to missing or incorrect data) by backend systems. This includes quality codes valid, verified, estimated, and derived / corrected.

HistoricalReadingList Object (*List*)

6860 A List element to hold HistoricalReading objects.

ProjectionReading Object (*BillingMeterReadingBase*)

6861 Contains values that forecast a future reading for the time or interval specified.

ProjectionReadingList Object (*List*)

6862 A List element to hold ProjectionReading objects.

TargetReading Object (*BillingMeterReadingBase*)

6863 Contains readings that specify a target or goal, such as a consumption target, to which billing incentives or other contractual ramifications may be associated.

TargetReadingList Object (*List*)

6864 A List element to hold TargetReading objects.

ServiceSupplier Object (*IdentifiedObject*)

6865 Organisation that provides services to Customers.

6873 ***email attribute (String32) [0..1]***

6874 E-mail address for this service supplier.

6875 ***phone attribute (String20) [0..1]***

6876 Human-readable phone number for this service supplier.

6877 ***providerID attribute (UInt32) [0..1]***

6878 Contains the IANA PEN for the commodity provider.

6879 ***web attribute (String42) [0..1]***

6880 Website URI address for this service supplier.

6881 **ServiceSupplierList Object (List)**

6882 A List element to hold ServiceSupplier objects.

6883 15.1.19 Prepayment Package

6884 Contains definitions related to storing and using payments.

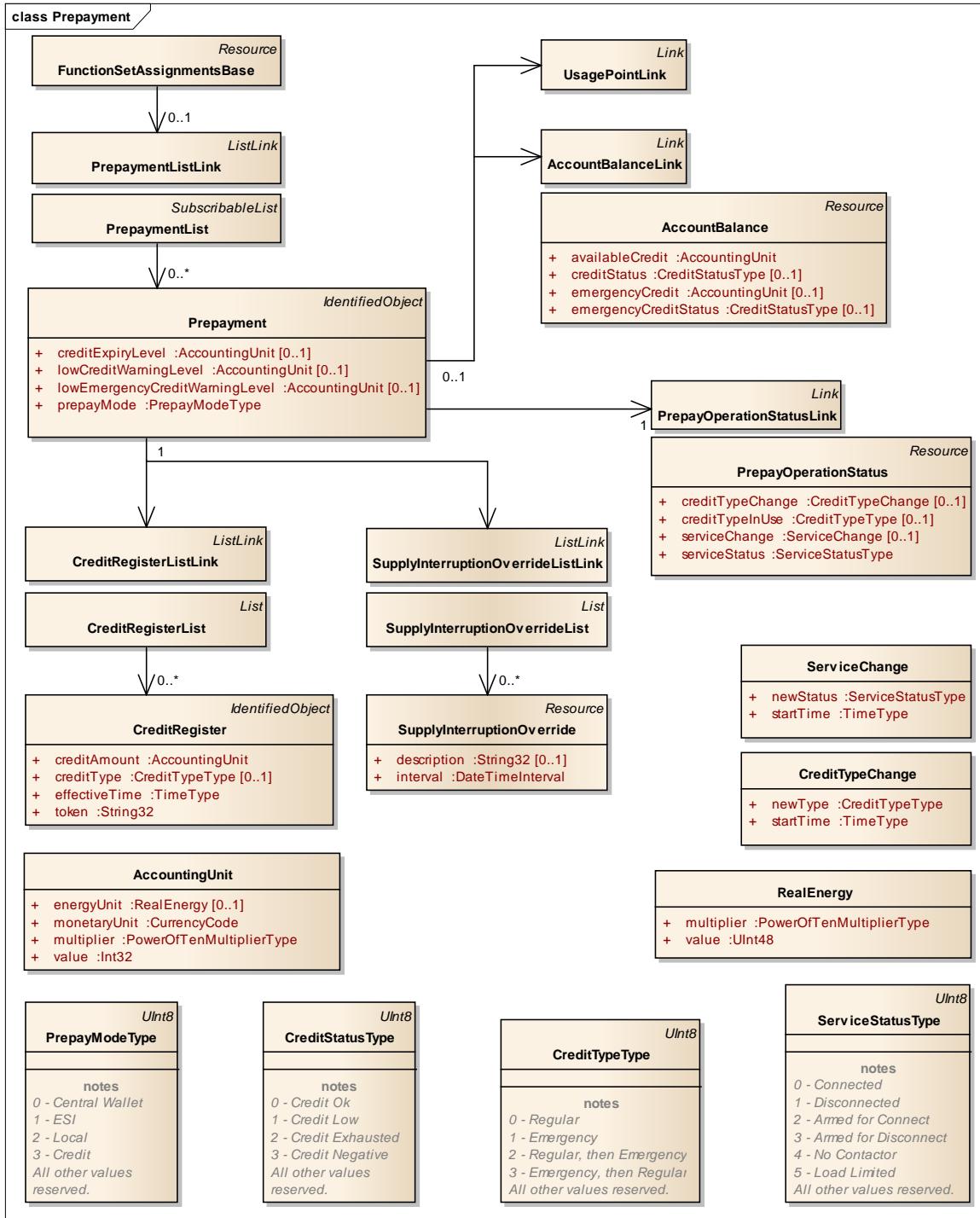


Figure 15-31: Prepayment

6885

6886

6887 **AccountBalance Object** (Resource)

6888 AccountBalance contains the regular credit and emergency credit balance for this given service or
6889 commodity prepay instance. It may also contain status information concerning the balance data.

6890 **availableCredit attribute** (AccountingUnit)

6891 AvailableCredit shows the balance of the sum of credits minus the sum of charges. In a Central Wallet
6892 mode this value may be passed down to the Prepayment server via an out-of-band mechanism. In Local or
6893 ESI modes, this value may be calculated based upon summation of CreditRegister transactions minus
6894 consumption charges calculated using Metering (and possibly Pricing) function set data. This value may
6895 be negative; for instance, if disconnection is prevented due to a Supply Interruption Override.

6896 **creditStatus attribute** (CreditStatusType) [0..1]

6897 CreditStatus identifies whether the present value of availableCredit is considered OK, low, exhausted, or
6898 negative.

6899 **emergencyCredit attribute** (AccountingUnit) [0..1]

6900 EmergencyCredit is the amount of credit still available for the given service or commodity prepayment
6901 instance. If both availableCredit and emergencyCredit are exhausted, then service will typically be
6902 disconnected.

6903 **emergencyCreditStatus attribute** (CreditStatusType) [0..1]

6904 EmergencyCreditStatus identifies whether the present value of emergencyCredit is considered OK, low,
6905 exhausted, or negative.

6906 **AccountingUnit Object** ()

6907 Unit for accounting; use either 'energyUnit' or 'currencyUnit' to specify the unit for 'value'.

6908 **energyUnit attribute** (RealEnergy) [0..1]

6909 Unit of service.

6910 **monetaryUnit attribute** (CurrencyCode)

6911 Unit of currency.

6912 **multiplier attribute** (PowerOfTenMultiplierType)

6913 Multiplier for the 'energyUnit' or 'monetaryUnit'.

6914 **value attribute** (Int32)

6915 Value of the monetary aspect

6916 **CreditRegister Object** (IdentifiedObject)

6917 CreditRegister instances define a credit-modifying transaction. Typically this would be a credit-
6918 adding transaction, but may be a subtracting transaction (perhaps in response to an out-of-band
6919 debt signal).

6920 **creditAmount attribute** (AccountingUnit)

6921 CreditAmount is the amount of credit being added by a particular CreditRegister transaction. Negative
6922 values indicate that credit is being subtracted.

6923 **creditType attribute** (CreditTypeType) [0..1]

6924 CreditType indicates whether the credit transaction applies to regular or emergency credit.

6925 **effectiveTime attribute** (TimeType)

6926 EffectiveTime identifies the time at which the credit transaction goes into effect. For credit addition
6927 transactions, this is typically the moment at which the transaction takes place. For credit subtraction

6928 transactions, (e.g., non-fuel debt recovery transactions initiated from a back-haul or ESI) this may be a
 6929 future time at which credit is deducted.

6930 *token attribute* (*String32*)

6931 Token is security data that authenticates the legitimacy of the transaction. The details of this token are not
 6932 defined by Smart Energy 2.0. How a Prepayment server handles this field is left as vendor specific
 6933 implementation or will be defined by one or more other standards.

6934 *CreditRegisterList Object* (*List*)

6935 A List element to hold CreditRegister objects.

6936 *Prepayment Object* (*IdentifiedObject*)

6937 Prepayment (inherited from CIM SDPAccountingFunction)

6938 *creditExpiryLevel attribute* (*AccountingUnit*) [0..1]

6939 CreditExpiryLevel is the set point for availableCredit at which the service level may be changed. The
 6940 typical value for this attribute is 0, regardless of whether the account balance is measured in a monetary
 6941 or commodity basis. The units for this attribute SHALL match the units used for availableCredit.

6942 *lowCreditWarningLevel attribute* (*AccountingUnit*) [0..1]

6943 LowCreditWarningLevel is the set point for availableCredit at which the creditStatus attribute in the
 6944 AccountBalance resource SHALL indicate that available credit is low. The units for this attribute SHALL
 6945 match the units used for availableCredit. Typically, this value is set by the service provider.

6946 *lowEmergencyCreditWarningLevel attribute* (*AccountingUnit*) [0..1]

6947 LowEmergencyCreditWarningLevel is the set point for emergencyCredit at which the creditStatus
 6948 attribute in the AccountBalance resource SHALL indicate that emergencycredit is low. The units for this
 6949 attribute SHALL match the units used for availableCredit. Typically, this value is set by the service
 6950 provider.

6951 *prepayMode attribute* (*PrepayModeType*)

6952 PrepayMode specifies whether the given Prepayment instance is operating in Credit, Central Wallet, ESI,
 6953 or Local prepayment mode. The Credit mode indicates that prepayment is not presently in effect. The
 6954 other modes are described in the Overview Section above.

6955 *PrepaymentList Object* (*SubscribableList*)

6956 A List element to hold Prepayment objects.

6957 *PrepayModeType Object* (*UInt8*)

6958 0 - Central Wallet

6959 1 - ESI

6960 2 - Local

6961 3 - Credit

6962 All other values reserved.

6963 *PrepayOperationStatus Object* (*Resource*)

6964 PrepayOperationStatus describes the status of the service or commodity being conditionally
 6965 controlled by the Prepayment function set.

6966 *creditTypeChange attribute* (*CreditTypeChange*) [0..1]

6967 CreditTypeChange is used to define a pending change of creditTypeInUse, which will activate at a

6968 specified time.

6969 ***creditTypeInUse attribute*** (*CreditTypeType*) [0..1]

6970 CreditTypeInUse identifies whether the present mode of operation is consuming regular credit or
6971 emergency credit.

6972 ***serviceChange attribute*** (*ServiceChange*) [0..1]

6973 ServiceChange is used to define a pending change of serviceStatus, which will activate at a specified
6974 time.

6975 ***serviceStatus attribute*** (*ServiceStatusType*)

6976 ServiceStatus identifies whether the service is connected or disconnected, or armed for connection or
6977 disconnection.

6978 **ServiceChange Object** ()

6979 Specifies a change to the service status.

6980 ***newStatus attribute*** (*ServiceStatusType*)

6981 The new service status, to take effect at the time specified by startTime

6982 ***startTime attribute*** (*TimeType*)

6983 The date/time when the change is to take effect.

6984 **SupplyInterruptionOverride Object** (Resource)

6985 SupplyInterruptionOverride: There may be periods of time when social, regulatory or other
6986 concerns mean that service should not be interrupted, even when available credit has been
6987 exhausted. Each Prepayment instance links to a List of SupplyInterruptionOverride instances.
6988 Each SupplyInterruptionOverride defines a contiguous period of time during which supply
6989 SHALL NOT be interrupted.

6990 ***description attribute*** (*String32*) [0..1]

6991 The description is a human readable text describing or naming the object.

6992 ***interval attribute*** (*DateTimeInterval*)

6993 Interval defines the period of time during which supply should not be interrupted.

6994 **SupplyInterruptionOverrideList Object** (List)

6995 A List element to hold SupplyInterruptionOverride objects.

6996 **CreditStatusType Object** (UInt8)

6997 0 - Credit Ok

6998 1 - Credit Low

6999 2 - Credit Exhausted

7000 3 - Credit Negative

7001 All other values reserved.

7002 **CreditTypeType Object** (UInt8)

7003 0 - Regular

7004 1 - Emergency

7005 2 - Regular, then Emergency

7006 3 - Emergency, then Regular

7007 All other values reserved.

CreditTypeChange Object ()

7009 Specifies a change to the credit type.

7010 **newType attribute (CreditTypeType)**

7011 The new credit type, to take effect at the time specified by startTime

7012 **startTime attribute (TimeType)**

7013 The date/time when the change is to take effect.

ServiceStatusType Object (UInt8)

7015 0 - Connected

7016 1 - Disconnected

7017 2 - Armed for Connect

7018 3 - Armed for Disconnect

7019 4 - No Contactor

7020 5 - Load Limited

7021 All other values reserved.

7022 15.1.20 **FlowReservation Package**

7023 Contains flow (charge) reservation model to allow fine-grained control of high-demand loads such as fast-
7024 charging batteries.

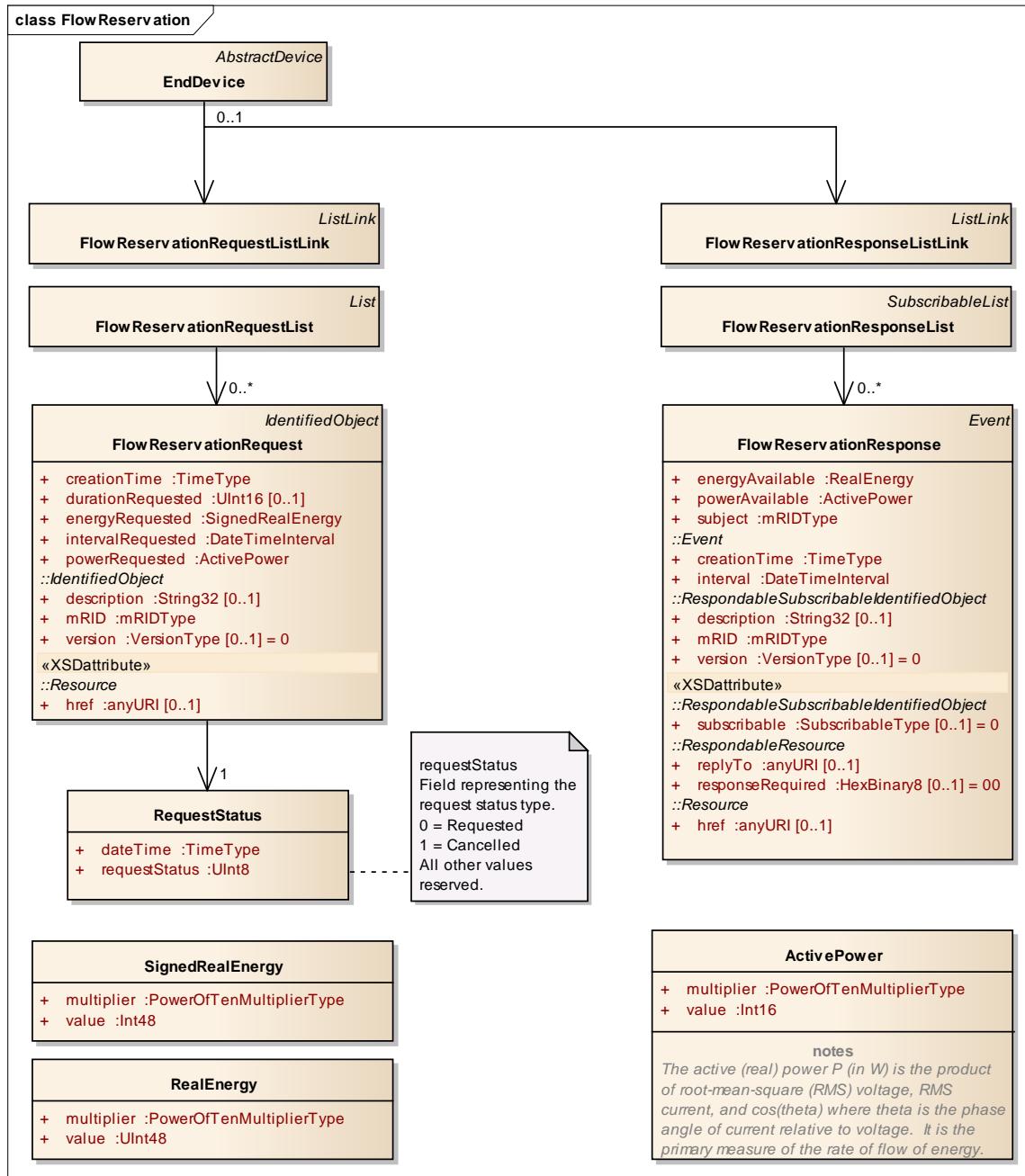


Figure 15-32: FlowReservation

7027 **RequestStatus Object ()**

7028 The RequestStatus object is used to indicate the current status of a Flow Reservation Request.

7029 **dateTime attribute (TimeType)**

7030 The **dateTime** attribute will provide a timestamp of when the request status was set. **dateTime** MUST be
 7031 set to the time at which the status change occurred, not a time in the future or past.

requestStatus attribute (UInt8)

7033 Field representing the request status type.

7034 0 = Requested

7035 1 = Cancelled

7036 All other values reserved.

AbstractFlowReservation Object (Event)

7038 Provides definition of FlowReservation elements in common between Requests and Responses.

FlowReservationRequest Object (IdentifiedObject)

7040 Used to request flow transactions. Client EndDevices submit a request for charging or
 7041 discharging from the server. The server creates an associated FlowReservationResponse
 7042 containing the charging parameters and interval to provide a lower aggregated demand at the
 7043 premises, or within a larger part of the distribution system.

creationTime attribute (TimeType)

7044 The time at which the request was created.

durationRequested attribute (UInt16) [0..1]

7045 A value that is calculated by the storage device that defines the minimum duration, in seconds, that it will
 7046 take to complete the actual flow transaction, including any ramp times and conditioning times, if
 7047 applicable.

energyRequested attribute (SignedRealEnergy)

7048 Indicates the total amount of energy, in Watt-Hours, requested to be transferred between the storage
 7049 device and the electric power system. Positive values indicate charging and negative values indicate
 7050 discharging. This sign convention is different than for the DER function where discharging is positive.
 7051 Note that the **energyRequestNow** attribute in the PowerStatus Object must always represent a charging
 7052 solution and it is not allowed to have a negative value.

intervalRequested attribute (DateTimeInterval)

7053 The time window during which the flow reservation is needed. For example, if an electric vehicle is set
 7054 with a 7:00 AM time charge is needed, and price drops to the lowest tier at 11:00 PM, then this window
 7055 would likely be from 11:00 PM until 7:00 AM.

powerRequested attribute (ActivePower)

7056 Indicates the sustained level of power, in Watts, that is requested. For charging this is calculated by the
 7057 storage device and it represents the charging system capability (which for an electric vehicle must also
 7058 account for any power limitations due to the EVSE control pilot). For discharging, a lower value than the
 7059 inverter capability can be used as a target.

FlowReservationRequestList Object (List)

7060 A List element to hold FlowReservationRequest objects.

FlowReservationResponse Object (Event)

7061 The server may modify the charging or discharging parameters and interval to provide a lower
 7062 aggregated demand at the premises, or within a larger part of the distribution system.

energyAvailable attribute (RealEnergy)

7071 Indicates the amount of energy available.

powerAvailable attribute (ActivePower)

7073 Indicates the amount of power available.

subject attribute (mRIDType)

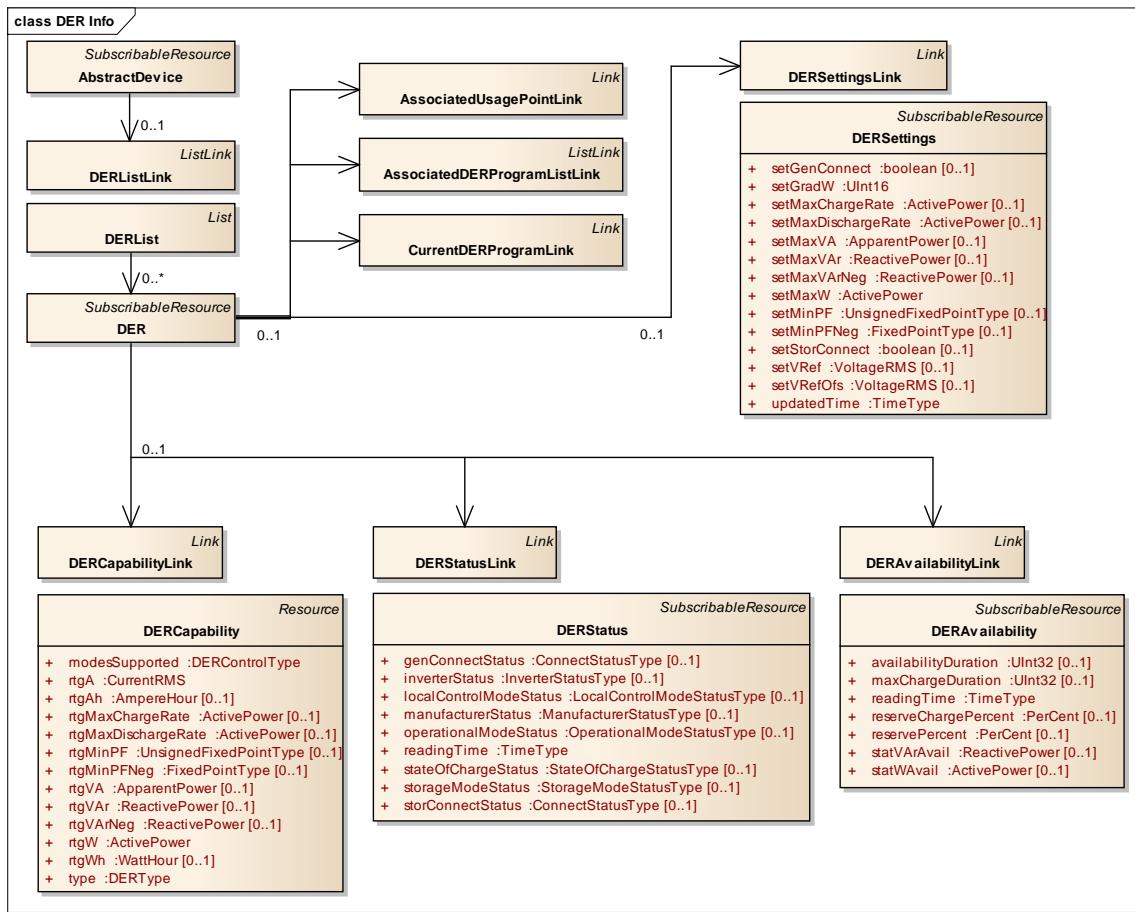
7075 The subject field provides a method to match the response with the originating event. It is populated with
7076 the mRID of the corresponding FlowReservationRequest object.

FlowReservationResponseList Object (SubscribableList)

7078 A List element to hold FlowReservationResponse objects.

15.1.21 DER Package

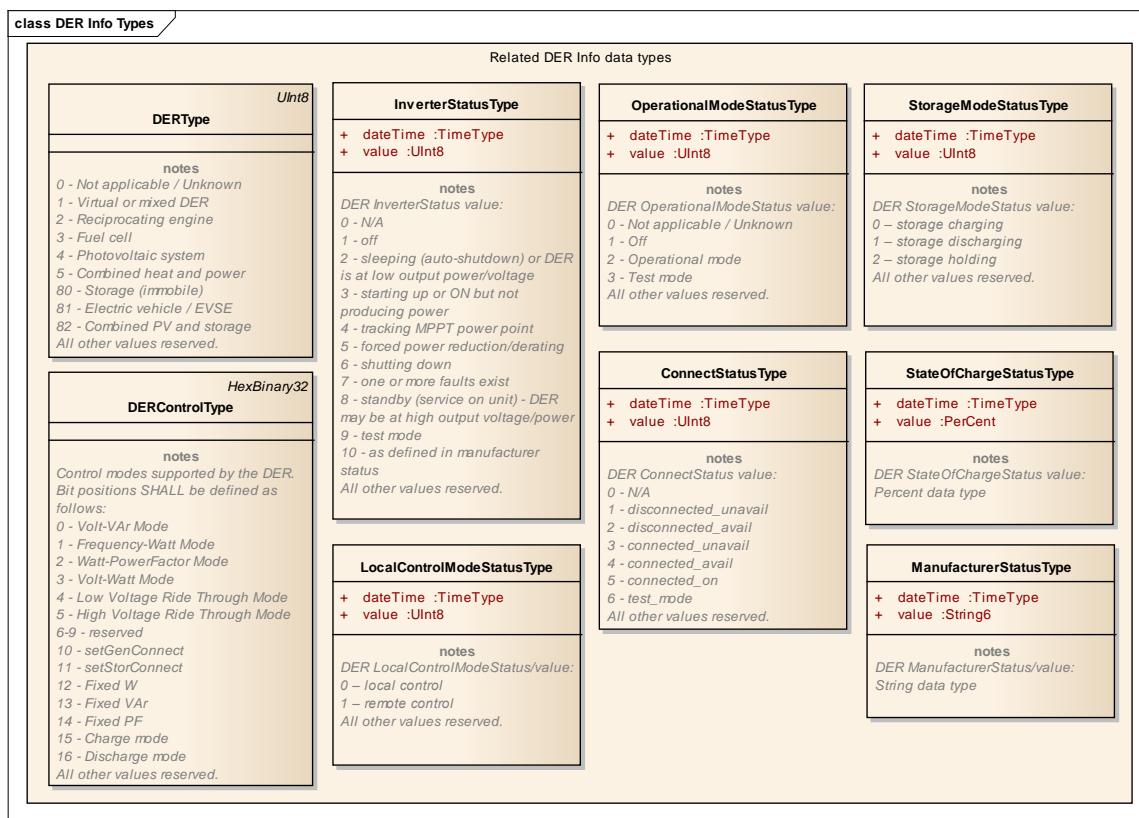
7080 Contains definitions related to allowing distributed energy resources to provide energy back to the grid.



7081

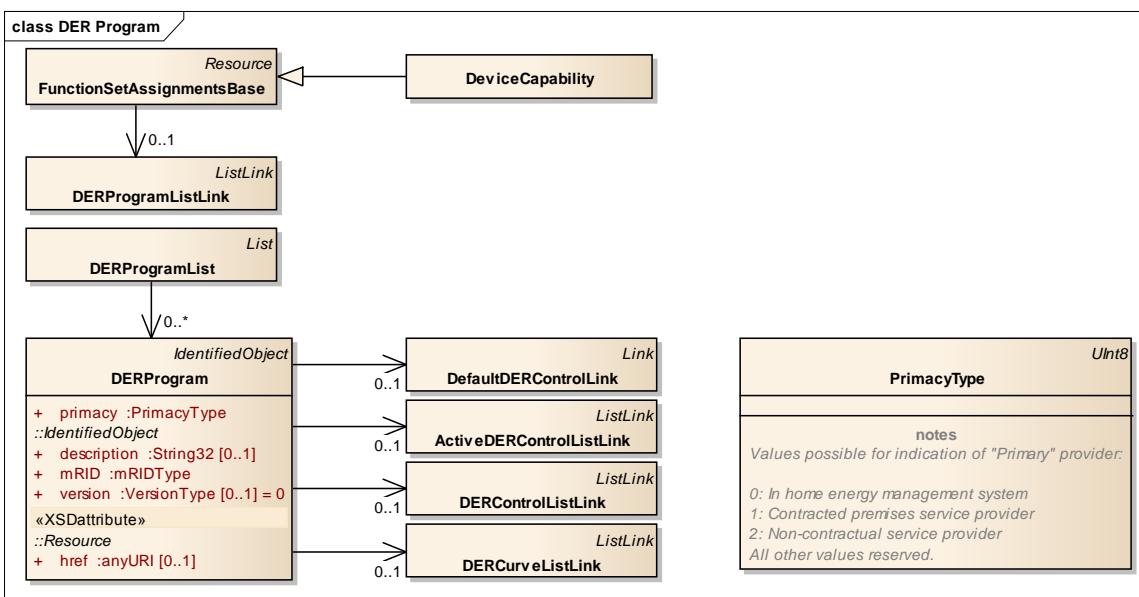
7082

Figure 15-33: DER Info



7083

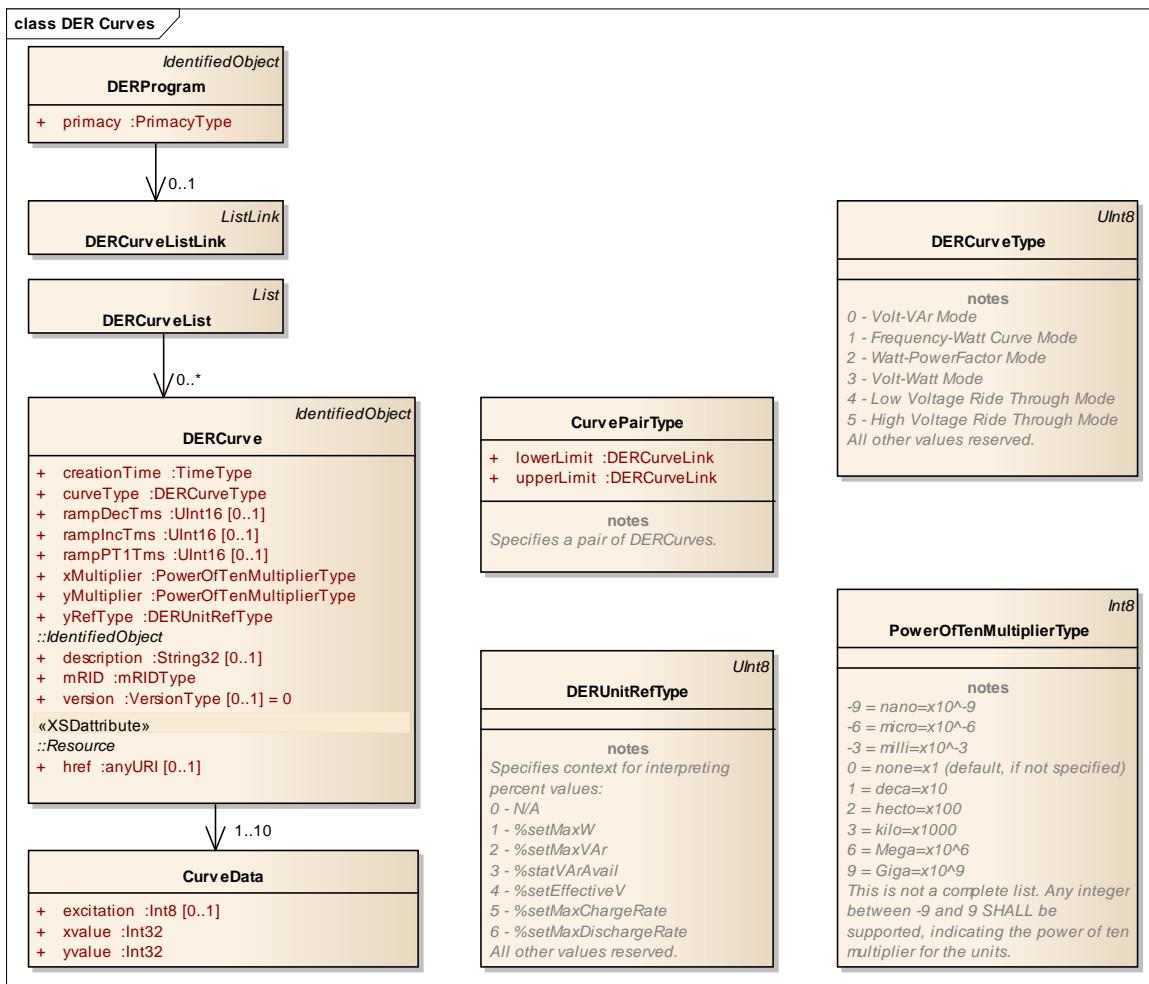
7084

Figure 15-34: DER Info Types

7085

7086

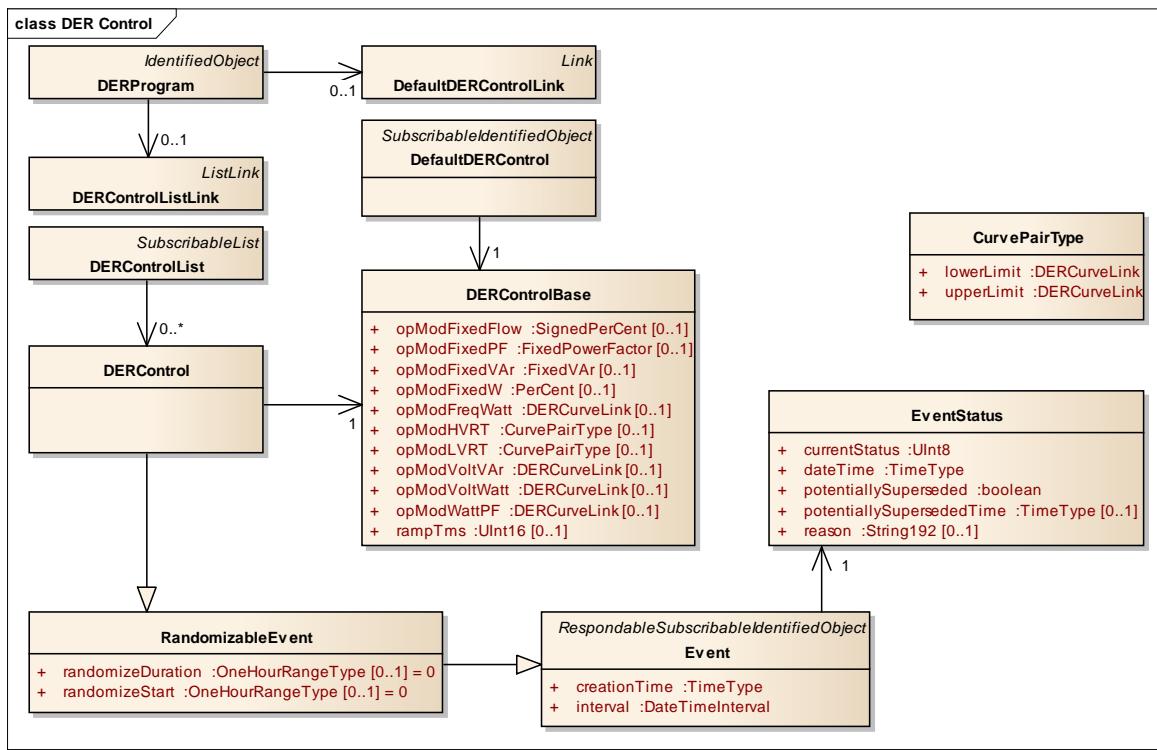
Figure 15-35: DER Program



7087

7088

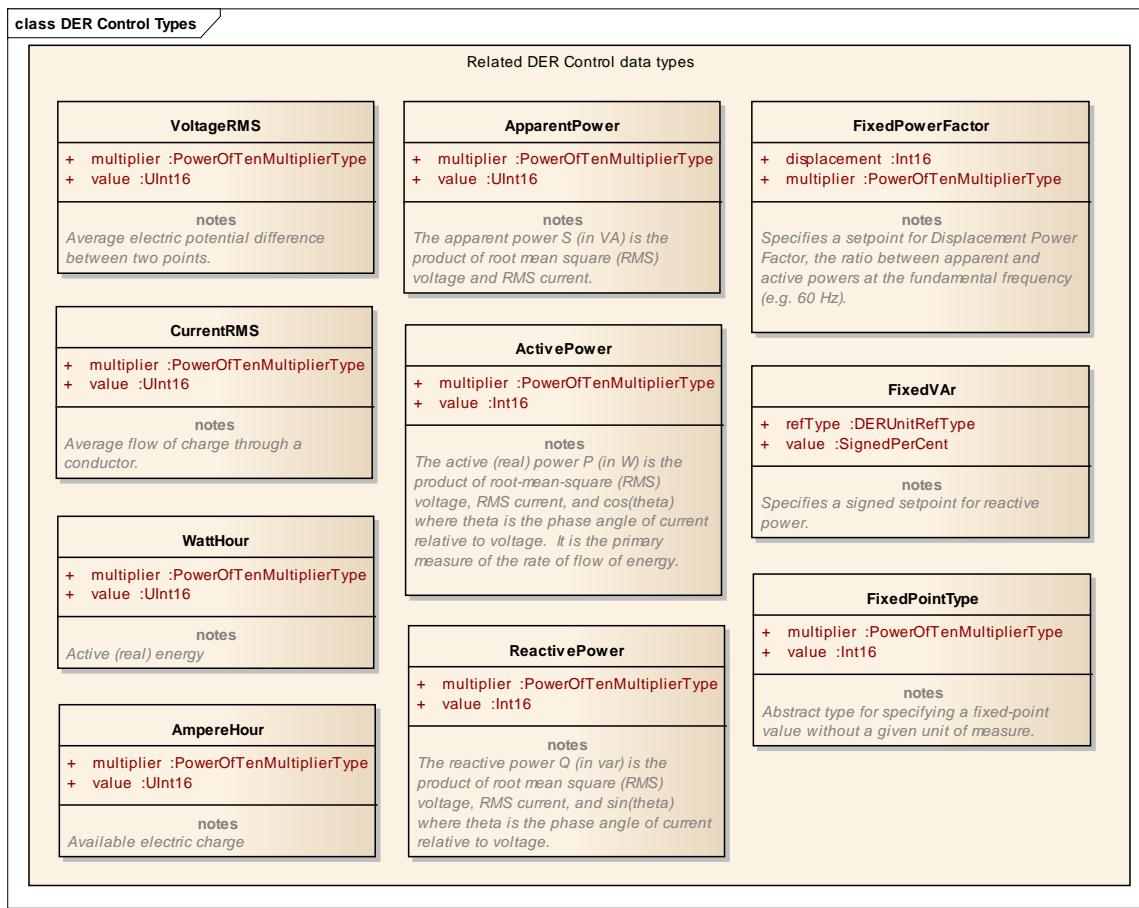
Figure 15-36: DER Curves



7089

7090

Figure 15-37: DER Control



7091

7092

Figure 15-38: DER Control Types7093 **DefaultDERControl Object** (SubscribableIdentifiedObject)

7094 Contains control mode information to be used if no active DERControl is found.

7095 **DER Object** (SubscribableResource)

7096 Contains links to DER resources.

7097 **DERList Object** (List)

7098 A List element to hold DER objects.

7099 **DERSettings Object** (SubscribableResource)

7100 Distributed energy resource settings

7101 **setGenConnect attribute (boolean) [0..1]**

7102 Set generator DER as connected (true) or disconnected (false).

7103 **setGradW attribute (UInt16)**7104 Set default rate of change (ramp rate) of active power output due to command or internal action, defined
7105 in %setWMax / second. Resolution is in hundredths of a percent/second and may be in the range 1 -
7106 20000. Interpreted as a percentage change in output capability limit per second when used as a default
7107 ramp rate.7108 **setMaxChargeRate attribute (ActivePower) [0..1]**

7109 Maximum rate of energy transfer received by the storage device, in Watts. Defaults to
 7110 rtgMaxChargeRate.

setMaxDischargeRate attribute (ActivePower) [0..1]

7112 Maximum rate of energy transfer delivered by the storage device, in Watts. Defaults to
 7113 rtgMaxDischargeRate.

setMaxVA attribute (ApparentPower) [0..1]

7114 Set limit for maximum apparent power capability of the DER (in VA). Defaults to rtgVA.

setMaxVAr attribute (ReactivePower) [0..1]

7115 Set limit for maximum reactive power delivered by the DER (in var). SHALL be a positive value <= rtgVAr (default).

setMaxVArNeg attribute (ReactivePower) [0..1]

7116 Set limit for maximum reactive power received by the DER (in var). If present, SHALL be a negative value >= rtgVArNeg (default). If absent, defaults to negative setMaxVAr.

setMaxW attribute (ActivePower)

7117 Set limit for maximum active power capability of the DER (in W). Defaults to rtgW.

setMinPF attribute (UnsignedFixedPointType) [0..1]

7118 Set minimum Power Factor displacement limit of the DER; positive value between 0.0 (typically > 0.7) and 1.0. SHALL be >= rtgMinPF (default).

setMinPFDeg attribute (FixedPointType) [0..1]

7119 Set minimum Power Factor displacement limit of the DER; negative value between 0.0 (typically < -0.7) and -0.9999. If present, SHALL be <= rtgMinPFDeg (default). If absent, defaults to negative setMinPF.

setStorConnect attribute (boolean) [0..1]

7120 Set storage DER as connected (true) or disconnected (false).

setVRef attribute (VoltageRMS) [0..1]

7121 The nominal AC voltage (RMS) at the utility's point of common coupling.

setVRefOfs attribute (VoltageRMS) [0..1]

7122 The nominal AC voltage (RMS) offset between the DER's electrical connection point and the utility's point of common coupling.

updatedTime attribute (TimeType)

7123 Specifies the time at which the DER information was last updated.

DERType Object (UInt8)

7124 0 - Not applicable / Unknown

7125 1 - Virtual or mixed DER

7126 2 - Reciprocating engine

7127 3 - Fuel cell

7128 4 - Photovoltaic system

7129 5 - Combined heat and power

7130 80 - Storage (immobile)

7147 81 - Electric vehicle / EVSE

7148 82 - Combined PV and storage

7149 All other values reserved.

DERAvailability Object (SubscribableResource)

7150 Indicates current reserve generation status

availabilityDuration attribute (UInt32) [0..1]

7151 Indicates number of seconds the DER will be able to deliver active power at the reservePercent level.

maxChargeDuration attribute (UInt32) [0..1]

7152 Indicates number of seconds the DER will be able to receive active power at the reserveChargePercent level.

readingTime attribute (TimeType)

7153 The timestamp when the DER availability was last updated.

reserveChargePercent attribute (PerCent) [0..1]

7154 Percent of continuous received active power (%setMaxChargeRate) that is estimated to be available in reserve.

reservePercent attribute (PerCent) [0..1]

7155 Percent of continuous delivered active power (%setMaxW) that is estimated to be available in reserve.

statVArAvail attribute (ReactivePower) [0..1]

7156 Estimated reserve reactive power, in var. Represents the lesser of received or delivered reactive power.

statWAval attribute (ActivePower) [0..1]

7157 Estimated reserve active power, in watts.

DERCapability Object (Resource)

7158 Distributed energy resource type and nameplate ratings.

modesSupported attribute (DERControlType)

7159 Bitmap indicating the DER Controls implemented by the device. See DERControlType for values.

rtgA attribute (CurrentRMS)

7160 Maximum continuous AC current capability of the DER, in Amperes (RMS).

rtgAh attribute (AmpereHour) [0..1]

7161 Usable energy storage capacity of the DER, in AmpHours.

rtgMaxChargeRate attribute (ActivePower) [0..1]

7162 Maximum rate of energy transfer received by the storage DER, in Watts.

rtgMaxDischargeRate attribute (ActivePower) [0..1]

7163 Maximum rate of energy transfer delivered by the storage DER, in Watts. Required for combined generation/storage DERs (e.g. DERType == 82).

rtgMinPF attribute (UnsignedFixedPointType) [0..1]

7164 Minimum Power Factor displacement capability of the DER; SHALL be a positive value between 0.0 (typically > 0.7) and 1.0. If absent, defaults to unity. (Unity power factor is considered unsigned.)

rtgMinPFneg attribute (FixedPointType) [0..1]

7165 Minimum Power Factor displacement capability of the DER; SHALL be a negative value between 0.0

7186 (typically < -0.7) and -0.9999. If absent, defaults to negative rtgMinPF. (Unity power factor is considered
 7187 unsigned.)

rtgVA attribute (ApparentPower) [0..1]

7189 Maximum continuous apparent power output capability of the DER, in VA.

rtgVAr attribute (ReactivePower) [0..1]

7191 Maximum continuous reactive power delivered by the DER, in var.

rtgVArNeg attribute (ReactivePower) [0..1]

7193 Maximum continuous reactive power received by the DER, in var. If absent, defaults to negative rtgVAr.

rtgW attribute (ActivePower)

7195 Maximum continuous active power output capability of the DER, in watts. Represents combined
 7196 generation plus storage output if DERType == 82.

rtgWh attribute (WattHour) [0..1]

7198 Maximum energy storage capacity of the DER, in WattHours.

type attribute (DERType)

7200 Type of DER; see DERType object

DERControlBase Object ()

7202 Distributed Energy Resource (DER) control values.

opModFixedFlow attribute (SignedPerCent) [0..1]

7203 The opModFixedFlow function specifies a requested charge or discharge mode setpoint,
 7204 in %setMaxChargeRate if negative value or %setMaxW or %setMaxDischargeRate if positive value (in
 7205 hundredths). SHALL be ignored if device is not a storage DER or setStorConnect is false.

opModFixedPF attribute (FixedPowerFactor) [0..1]

7206 The opModFixedPF function specifies a requested fixed Power Factor (PF) setting, consisting of a signed
 7207 displacement value. The PF sign (which SHALL be interpreted according to the EEI convention, where
 7208 unity PF is considered unsigned) indicates the direction of reactive power flow. The actual displacement
 7209 SHALL be within the limits established by setMinPF and setMinPfneg. If issued simultaneously with
 7210 other reactive power controls (e.g. opModFixedVAr) the control resulting in least var magnitude takes
 7211 precedence.

opModFixedVAr attribute (FixedVAr) [0..1]

7212 The opModFixedVAr function specifies the delivered or received reactive power limit setpoint. The
 7213 context for the limit value is determined by refType and SHALL be one of %setMaxW, %setMaxVAr,
 7214 or %statVArAvail. If issued simultaneously with other reactive power controls (e.g. opModFixedPF) the
 7215 control resulting in least var magnitude takes precedence.

opModFixedW attribute (PerCent) [0..1]

7216 The opModFixedW function sets the maximum active power generation level at the electrical coupling
 7217 point as a percentage of set capacity (%setMaxW, in hundredths). This limitation may be met e.g. by
 7218 reducing PV output or by using excess PV output to charge associated storage.

opModFreqWatt attribute (DERCurveLink) [0..1]

7219 Specify DERCurveLink for curve type == 1. The Frequency-Watt function limits active power
 7220 generation or consumption when the line frequency deviates from nominal by a specified amount. The
 7221 Frequency-Watt curve is specified as an array of Frequency-Watt pairs that are interpolated into a
 7222 piecewise linear function with hysteresis. The x value of each pair specifies a frequency in Hz. The y

7228 value specifies a corresponding active power output in %setMaxW * 100 (0 - 10000) (hundredths of a
 7229 percent).

opModHVRT attribute (CurvePairType) [0..1]

7230 Specify curve pair for curve type == 5. The High Voltage Ride-Through (HVRT) function is specified by
 7231 one or two duration-volt curves that define the operating region under high voltage conditions. Each
 7232 HVRT curve is specified by an array of duration-volt pairs that will be interpolated into a piecewise linear
 7233 function that defines an operating region. The x value of each pair specifies a duration (time at a given
 7234 voltage in seconds). The y value of each pair specifies an effective percent voltage, defined as (100% *
 7235 (locally measured voltage - setVRefOfs) / setVRef), in hundredths of a percent. The upperLimit curve
 7236 delineates the "must disconnect" region and SHALL be defined for this control to be active. The
 7237 (optional) lowerLimit curve delineates the "must remain connected" region. If the "must remain
 7238 connected" curve is not specified, it is assumed to be the same as the "must disconnect" curve.
 7239

opModLVRT attribute (CurvePairType) [0..1]

7240 Specify curve pair for curve type == 4. The Low Voltage Ride-Through (LVRT) function is specified by
 7241 one or two duration-volt curves that define the operating region under low voltage conditions. Each
 7242 LVRT curve is specified by an array of duration-volt pairs that will be interpolated into a piecewise linear
 7243 function that defines an operating region. The x value of each pair specifies a duration (time at a given
 7244 voltage in seconds). The y value of each pair specifies an effective percent voltage, defined as (100% *
 7245 (locally measured voltage - setVRefOfs) / setVRef), in hundredths of a percent. The lowerLimit curve
 7246 delineates the "must disconnect" region and SHALL be defined for this control to be active. The
 7247 (optional) upperLimit curve delineates the "must remain connected" region. If the "must remain
 7248 connected" curve is not specified, it is assumed to be the same as the "must disconnect" curve.
 7249

opModVoltVAr attribute (DERCurveLink) [0..1]

7250 Specify DERCurveLink for curve type == 0. The static Volt-VAr function provides over- or under-
 7251 excited VAr compensation as a function of measured voltage. The Volt-VAr curve is specified as an array
 7252 of Volt-VAr pairs that are interpolated into a piecewise linear function with hysteresis. The x value of
 7253 each pair specifies an effective percent voltage, defined as (100% * (locally measured voltage -
 7254 setVRefOfs) / setVRef) and SHOULD support a domain of at least 0 - 13500 (in hundredths of a percent).
 7255 The y value specifies a target VAr output interpreted as SignedPerCent (10000 to -10000). The meaning
 7256 of the y value is determined by yRefType and must be one of %setMaxW, %setMaxVAr,
 7257 or %statVArAvail, all in hundredths of a percent.
 7258

opModVoltWatt attribute (DERCurveLink) [0..1]

7259 Specify DERCurveLink for curve type == 3. The Volt-Watt reduces active power output as a function of
 7260 measured voltage. The Volt-Watt curve is specified as an array of Volt-Watt pairs that are interpolated
 7261 into a piecewise linear function with hysteresis. The x value of each pair specifies an effective percent
 7262 voltage, defined as (100% * (locally measured voltage - setVRefOfs) / setVRef) and SHOULD support a
 7263 domain of at least 0 - 13500 (in hundredths of a percent). The y value specifies an active power output
 7264 in %setMaxW, (0 - 10000) in hundredths of a percent.
 7265

opModWattPF attribute (DERCurveLink) [0..1]

7266 Specify DERCurveLink for curve type == 2. The Watt-PF function varies Power Factor (PF) as a
 7267 function of delivered active power. The Watt-PF curve is specified as an array of Watt-PF coordinates
 7268 that are interpolated into a piecewise linear function with hysteresis. The x value of each pair specifies a
 7269 watt setting in %setMaxW, (0 - 10000) in hundredths of a percent. The PF output setting is a signed
 7270 displacement in y value (PF sign SHALL be interpreted according to the EEI convention, where unity PF
 7271 is considered unsigned). These settings are not expected to be updated very often during the life of the
 7272

7273 installation, therefore only a single curve is required. If issued simultaneously with other reactive power
 7274 controls (e.g. opModFixedPF) the control resulting in least var magnitude takes precedence.

7275 ***rampTms attribute (UInt16) [0..1]***

7276 Requested ramp time, in hundredths of a second, for the device to transition from the current DERControl
 7277 mode setting(s) to the new mode setting(s). If absent, use default ramp rate (setWGrad). Resolution is
 7278 1/100 sec.

7279 ***DERControl Object*** (RandomizableEvent)

7280 Distributed Energy Resource (DER) time/event-based control.

7281 ***DERControlList Object*** (SubscribableList)

7282 A List element to hold DERControl objects.

7283 ***DERControlType Object*** (HexBinary32)

7284 Control modes supported by the DER. Bit positions SHALL be defined as follows:

7285 0 - Volt-VAr Mode

7286 1 - Frequency-Watt Mode

7287 2 - Watt-PowerFactor Mode

7288 3 - Volt-Watt Mode

7289 4 - Low Voltage Ride Through Mode

7290 5 - High Voltage Ride Through Mode

7291 6-9 - reserved

7292 10 - setGenConnect

7293 11 - setStorConnect

7294 12 - Fixed W

7295 13 - Fixed VAr

7296 14 - Fixed PF

7297 15 - Charge mode

7298 16 - Discharge mode

7299 All other values reserved.

7300 ***DERCurve Object*** (IdentifiedObject)

7301 DER related curves such as Volt-VAr mode curves. Relationship between an independent
 7302 variable (X-axis) and one or two dependent variables (Y-axis and excitation).

7303 ***creationTime attribute (TimeType)***

7304 The time at which the object was created.

7305 ***curveType attribute (DERCurveType)***

7306 Specifies the associated curve-based control mode.

7307 ***rampDecTms attribute (UInt16) [0..1]***

7308 Decreasing ramp rate, interpreted as a percentage change in output capability limit per second
 7309 (e.g. %setMaxW / sec). Resolution is in hundredths of a percent/second and may be in the range 1 -

7310 20000. If absent, ramp rate defaults to setWGrad.

rampIncTms attribute (UInt16) [0..1]

7312 Increasing ramp rate, interpreted as a percentage change in output capability limit per second
 7313 (e.g. %setMaxW / sec). Resolution is in hundredths of a percent/second and may be in the range 1 -
 7314 20000. If absent, ramp rate defaults to rampDecTms.

rampPT1Tms attribute (UInt16) [0..1]

7316 The configuration parameter for a low-pass filter, PT1 is a time, in hundredths of a second, in which the
 7317 filter will settle to 95% of a step change in the input value. Resolution is 1/100 sec.

xMultiplier attribute (PowerOfTenMultiplierType)

7319 Exponent for X-axis value.

yMultiplier attribute (PowerOfTenMultiplierType)

7321 Exponent for Y-axis value.

yRefType attribute (DERUnitRefType)

7323 The Y-axis units context.

CurrentDERProgramLink Object (Link)

7325 SHALL contain a Link to an instance of DERProgram. If present, this is the DERProgram
 7326 containing the currently active DERControl.

DERCurveList Object (List)

7328 A List element to hold DERCurve objects.

CurveData Object ()

7330 Data point values for defining a curve or schedule

excitation attribute (Int8) [0..1]

7332 To be included with curve type 2 (Watt-PowerFactor), specifies the excitation of the power factor.

7333 -1 = under-excited

7334 1 = over-excited

xvalue attribute (Int32)

7336 The data value of the X-axis (independent) variable, depending on the curve type. See definitions in
 7337 DERControlBase for further information.

yvalue attribute (Int32)

7339 The data value of the Y-axis (dependent) variable, depending on the curve type. See definitions in
 7340 DERControlBase for further information.

DERCurveType Object (UInt8)

7341 0 - Volt-VAr Mode

7343 1 - Frequency-Watt Curve Mode

7344 2 - Watt-PowerFactor Mode

7345 3 - Volt-Watt Mode

7346 4 - Low Voltage Ride Through Mode

7347 5 - High Voltage Ride Through Mode

- 7348 All other values reserved.
- 7349 **CurvePairType Object ()**
7350 Specifies a pair of DERCurves.
- 7351 ***lowerLimit attribute (DERCurveLink)***
7352 DERCurveLink for lower bound of operating region.
- 7353 ***upperLimit attribute (DERCurveLink)***
7354 DERCurveLink for upper bound of operating region.
- 7355 **DERProgram Object (IdentifiedObject)**
7356 Distributed Energy Resource program.
- 7357 ***primacy attribute (PrimacyType)***
7358 Indicates the relative primacy of the provider of this Program.
- 7359 **DERProgramList Object (List)**
7360 A List element to hold DERProgram objects.
- 7361 **DERStatus Object (SubscribableResource)**
7362 DER status information.
- 7363 ***genConnectStatus attribute (ConnectStatusType) [0..1]***
7364 Connect/status value for generator DER.
- 7365 See ConnectStatusType for values.
- 7366 ***inverterStatus attribute (InverterStatusType) [0..1]***
7367 DER InverterStatus/value.
- 7368 See InverterStatusType for values.
- 7369 ***localControlModeStatus attribute (LocalControlModeStatusType) [0..1]***
7370 The local control mode status.
- 7371 See LocalControlModeStatusType for values.
- 7372 ***manufacturerStatus attribute (ManufacturerStatusType) [0..1]***
7373 Manufacturer status code.
- 7374 ***operationalModeStatus attribute (OperationalModeStatusType) [0..1]***
7375 Operational mode currently in use.
- 7376 See OperationalModeStatusType for values.
- 7377 ***readingTime attribute (TimeType)***
7378 The timestamp when the current status was last updated.
- 7379 ***stateOfChargeStatus attribute (StateOfChargeStatusType) [0..1]***
7380 State of charge status.
- 7381 See StateOfChargeStatusType for values.
- 7382 ***storageModeStatus attribute (StorageModeStatusType) [0..1]***
7383 Storage mode status.
- 7384 See StorageModeStatusType for values.

7385 ***storConnectStatus attribute*** (*ConnectStatusType*) [0..1]

7386 Connect/status value for storage DER.

7387 See *ConnectStatusType* for values.

7388 ***DERUnitRefType Object*** (*UInt8*)

7389 Specifies context for interpreting percent values:

7390 0 - N/A

7391 1 - %setMaxW

7392 2 - %setMaxVAr

7393 3 - %statVArAvail

7394 4 - %setEffectiveV

7395 5 - %setMaxChargeRate

7396 6 - %setMaxDischargeRate

7397 All other values reserved.

7398 ***CurrentRMS Object*** ()

7399 Average flow of charge through a conductor.

7400 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7401 Specifies exponent of value.

7402 ***value attribute*** (*UInt16*)

7403 Value in amperes RMS (uom 5)

7404 ***FixedPointType Object*** ()

7405 Abstract type for specifying a fixed-point value without a given unit of measure.

7406 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7407 Specifies exponent of uom.

7408 ***value attribute*** (*Int16*)

7409 Dimensionless value

7410 ***UnsignedFixedPointType Object*** ()

7411 Abstract type for specifying an unsigned fixed-point value without a given unit of measure.

7412 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7413 Specifies exponent of uom.

7414 ***value attribute*** (*UInt16*)

7415 Dimensionless value

7416 ***ActivePower Object*** ()

7417 The active (real) power P (in W) is the product of root-mean-square (RMS) voltage, RMS current, and cos(theta) where theta is the phase angle of current relative to voltage. It is the primary measure of the rate of flow of energy.

7420 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7421 Specifies exponent for uom.

7422 ***value attribute*** (*Int16*)

7423 Value in watts (uom 38)

7424 **AmpereHour Object** ()

7425 Available electric charge

7426 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7427 Specifies exponent of uom.

7428 ***value attribute*** (*UInt16*)

7429 Value in ampere-hours (uom 106)

7430 **ApparentPower Object** ()

7431 The apparent power S (in VA) is the product of root mean square (RMS) voltage and RMS current.

7433 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7434 Specifies exponent of uom.

7435 ***value attribute*** (*UInt16*)

7436 Value in volt-amperes (uom 61)

7437 **ReactivePower Object** ()

7438 The reactive power Q (in var) is the product of root mean square (RMS) voltage, RMS current, and sin(theta) where theta is the phase angle of current relative to voltage.

7440 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7441 Specifies exponent of uom.

7442 ***value attribute*** (*Int16*)

7443 Value in volt-amperes reactive (var) (uom 63)

7444 **FixedPowerFactor Object** ()

7445 Specifies a setpoint for Displacement Power Factor, the ratio between apparent and active powers at the fundamental frequency (e.g. 60 Hz).

7447 ***displacement attribute*** (*Int16*)

7448 Significand of a signed value of cos(theta) between -0.9999 and 1.0. E.g. a value of -0.95 may be specified as a displacement of -950 and a multiplier of -3. Sign SHALL be interpreted according to the EEI convention.

7451 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7452 Specifies exponent of 'displacement'.

7453 **FixedVAr Object** ()

7454 Specifies a signed setpoint for reactive power.

7455 ***refType attribute*** (*DERUnitRefType*)

7456 Indicates whether to interpret 'value' as %setMaxVAr or %statVArAvail.

7457 ***value attribute*** (*SignedPerCent*)

7458 Specify a signed setpoint for reactive power in % (see 'refType' for context).

7459 **WattHour Object** ()

7460 Active (real) energy

7461 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7462 Specifies exponent of uom.

7463 ***value attribute*** (*UInt16*)

7464 Value in watt-hours (uom 72)

7465 **VoltageRMS Object** ()

7466 Average electric potential difference between two points.

7467 ***multiplier attribute*** (*PowerOfTenMultiplierType*)

7468 Specifies exponent of uom.

7469 ***value attribute*** (*UInt16*)

7470 Value in volts RMS (uom 29)

7471 **ConnectStatusType Object** ()

7472 DER ConnectStatus value:

7473 0 - N/A

7474 1 - disconnected_unavail

7475 2 - disconnected_avail

7476 3 - connected_unavail

7477 4 - connected_avail

7478 5 - connected_on

7479 6 - test_mode

7480 All other values reserved.

7481 ***dateTime attribute*** (*TimeType*)

7482 The date and time at which the state applied.

7483 ***value attribute*** (*UInt8*)

7484 The value indicating the state.

7485 **InverterStatusType Object** ()

7486 DER InverterStatus value:

7487 0 - N/A

7488 1 - off

7489 2 - sleeping (auto-shutdown) or DER is at low output power/voltage

7490 3 - starting up or ON but not producing power

7491 4 - tracking MPPT power point

7492 5 - forced power reduction/derating

7493 6 - shutting down

7494 7 - one or more faults exist

7495 8 - standby (service on unit) - DER may be at high output voltage/power

7496 9 - test mode

7497 10 - as defined in manufacturer status

7498 All other values reserved.

dateTime attribute (TimeType)

7500 The date and time at which the state applied.

value attribute (UInt8)

7502 The value indicating the state.

LocalControlModeStatusType Object ()

7503 DER LocalControlModeStatus/value:

7505 0 – local control

7506 1 – remote control

7507 All other values reserved.

dateTime attribute (TimeType)

7509 The date and time at which the state applied.

value attribute (UInt8)

7511 The value indicating the state.

ManufacturerStatusType Object ()

7513 DER ManufacturerStatus/value: String data type

dateTime attribute (TimeType)

7515 The date and time at which the state applied.

value attribute (String6)

7517 The value indicating the state.

OperationalModeStatusType Object ()

7519 DER OperationalModeStatus value:

7520 0 - Not applicable / Unknown

7521 1 - Off

7522 2 - Operational mode

7523 3 - Test mode

7524 All other values reserved.

dateTime attribute (TimeType)

7526 The date and time at which the state applied.

value attribute (UInt8)

7528 The value indicating the state.

StateOfChargeStatusType Object ()

7530 DER StateOfChargeStatus value: Percent data type

dateTime attribute (TimeType)

7532 The date and time at which the state applied.

7533 ***value attribute*** (*PerCent*)

7534 The value indicating the state.

7535 **StorageModeStatusType Object** ()

7536 DER StorageModeStatus value:

7537 0 – storage charging

7538 1 – storage discharging

7539 2 – storage holding

7540 All other values reserved.

7541 ***dateTime attribute*** (*TimeType*)

7542 The date and time at which the state applied.

7543 ***value attribute*** (*UInt8*)

7544 The value indicating the state.

7545 15.1.22 **Links Package**

7546 Contains definitions of Link specializations used to require certain associations.

7547 **AccountBalanceLink Object** (Link)

7548 SHALL contain a Link to an instance of AccountBalance.

7549 **ActiveBillingPeriodListLink Object** (ListLink)

7550 SHALL contain a Link to a List of active BillingPeriod instances.

7551 **ActiveCreditRegisterListLink Object** (ListLink)

7552 SHALL contain a Link to a List of active CreditRegister instances.

7553 **ActiveDERControlListLink Object** (ListLink)

7554 SHALL contain a Link to a List of active DERControl instances.

7555 **ActiveEndDeviceControlListLink Object** (ListLink)

7556 SHALL contain a Link to a List of active EndDeviceControl instances.

7557 **ActiveFlowReservationListLink Object** (ListLink)

7558 SHALL contain a Link to a List of active FlowReservation instances.

7559 **ActiveProjectionReadingListLink Object** (ListLink)

7560 SHALL contain a Link to a List of active ProjectionReading instances.

7561 **ActiveSupplyInterruptionOverrideListLink Object** (ListLink)

7562 SHALL contain a Link to a List of active SupplyInterruptionOverride instances.

7563 **ActiveTargetReadingListLink Object** (ListLink)

7564 SHALL contain a Link to a List of active TargetReading instances.

7565 **ActiveTextMessageListLink Object** (ListLink)

7566 SHALL contain a Link to a List of active TextMessage instances.

7567 **ActiveTimeTariffIntervalListLink Object** (ListLink)

7568 SHALL contain a Link to a List of active TimeTariffInterval instances.

7569 **AssociatedDERProgramListLink Object** (ListLink)

7570 SHALL contain a Link to a List of DERPrograms having the DERControl(s) for this DER.

7571 AssociatedUsagePointLink Object (Link)

7572 SHALL contain a Link to an instance of UsagePoint. If present, this is the submeter that
7573 monitors the DER output.

7574 BillingPeriodListLink Object (ListLink)

7575 SHALL contain a Link to a List of BillingPeriod instances.

7576 BillingReadingListLink Object (ListLink)

7577 SHALL contain a Link to a List of BillingReading instances.

7578 BillingReadingSetListLink Object (ListLink)

7579 SHALL contain a Link to a List of BillingReadingSet instances.

7580 ConfigurationLink Object (Link)

7581 SHALL contain a Link to an instance of Configuration.

7582 ConsumptionTariffIntervalListLink Object (ListLink)

7583 SHALL contain a Link to a List of ConsumptionTariffInterval instances.

7584 CreditRegisterListLink Object (ListLink)

7585 SHALL contain a Link to a List of CreditRegister instances.

7586 CustomerAccountLink Object (Link)

7587 SHALL contain a Link to an instance of CustomerAccount.

7588 CustomerAccountListLink Object (ListLink)

7589 SHALL contain a Link to a List of CustomerAccount instances.

7590 CustomerAgreementListLink Object (ListLink)

7591 SHALL contain a Link to a List of CustomerAgreement instances.

7592 DemandResponseProgramLink Object (Link)

7593 SHALL contain a Link to an instance of DemandResponseProgram.

7594 DemandResponseProgramListLink Object (ListLink)

7595 SHALL contain a Link to a List of DemandResponseProgram instances.

7596 DERAvailabilityLink Object (Link)

7597 SHALL contain a Link to an instance of DERAvailability.

7598 DERCapabilityLink Object (Link)

7599 SHALL contain a Link to an instance of DERCapability.

7600 DefaultDERControlLink Object (Link)

7601 SHALL contain a Link to an instance of DefaultDERControl. This is the default mode of the
7602 DER which MAY be overridden by DERControl events.

7603 DERControlListLink Object (ListLink)

7604 SHALL contain a Link to a List of DERControl instances.

7605 DERCurveLink Object (Link)

7606 SHALL contain a Link to an instance of DERCurve.

7607 DERCurveListLink Object (ListLink)

7608 SHALL contain a Link to a List of DERCurve instances.

- 7609 **DERLink Object** (Link)
7610 SHALL contain a Link to an instance of DER.
- 7611 **DERListLink Object** (ListLink)
7612 SHALL contain a Link to a List of DER instances.
- 7613 **DERProgramLink Object** (Link)
7614 SHALL contain a Link to an instance of DERProgram.
- 7615 **DERProgramListLink Object** (ListLink)
7616 SHALL contain a Link to a List of DERProgram instances.
- 7617 **DERSettingsLink Object** (Link)
7618 SHALL contain a Link to an instance of DERSettings.
- 7619 **DERStatusLink Object** (Link)
7620 SHALL contain a Link to an instance of DERStatus.
- 7621 **DeviceCapabilityLink Object** (Link)
7622 SHALL contain a Link to an instance of DeviceCapability.
- 7623 **DeviceInformationLink Object** (Link)
7624 SHALL contain a Link to an instance of DeviceInformation.
- 7625 **DeviceStatusLink Object** (Link)
7626 SHALL contain a Link to an instance of DeviceStatus.
- 7627 **EndDeviceControlListLink Object** (ListLink)
7628 SHALL contain a Link to a List of EndDeviceControl instances.
- 7629 **EndDeviceLink Object** (Link)
7630 SHALL contain a Link to an instance of EndDevice.
- 7631 **EndDeviceListLink Object** (ListLink)
7632 SHALL contain a Link to a List of EndDevice instances.
- 7633 **FileLink Object** (Link)
7634 This element MUST be set to the URI of the most recent File being loaded/activated by the LD.
7635 In the case of file status 0, this element MUST be omitted.
- 7636 **FileListLink Object** (ListLink)
7637 SHALL contain a Link to a List of File instances.
- 7638 **FileStatusLink Object** (Link)
7639 SHALL contain a Link to an instance of FileStatus.
- 7640 **FlowReservationRequestListLink Object** (ListLink)
7641 SHALL contain a Link to a List of FlowReservationRequest instances.
- 7642 **FlowReservationResponseListLink Object** (ListLink)
7643 SHALL contain a Link to a List of FlowReservationResponse instances.
- 7644 **FunctionSetAssignmentsListLink Object** (ListLink)
7645 SHALL contain a Link to a List of FunctionSetAssignments instances.
- 7646 **HistoricalReadingListLink Object** (ListLink)
7647 SHALL contain a Link to a List of HistoricalReading instances.

- 7648 **IPAddrListLink Object** (ListLink)
7649 SHALL contain a Link to a List of IPAddr instances.
- 7650 **IPInterfaceListLink Object** (ListLink)
7651 SHALL contain a Link to a List of IPInterface instances.
- 7652 **LLInterfaceListLink Object** (ListLink)
7653 SHALL contain a Link to a List of LLInterface instances.
- 7654 **LoadShedAvailabilityLink Object** (Link)
7655 SHALL contain a Link to an instance of LoadShedAvailability.
- 7656 **LogEventListLink Object** (ListLink)
7657 SHALL contain a Link to a List of LogEvent instances.
- 7658 **MessagingProgramListLink Object** (ListLink)
7659 SHALL contain a Link to a List of MessagingProgram instances.
- 7660 **MeterReadingLink Object** (Link)
7661 SHALL contain a Link to an instance of MeterReading.
- 7662 **MeterReadingListLink Object** (ListLink)
7663 SHALL contain a Link to a List of MeterReading instances.
- 7664 **MirrorUsagePointListLink Object** (ListLink)
7665 SHALL contain a Link to a List of MirrorUsagePoint instances.
- 7666 **NeighborListLink Object** (ListLink)
7667 SHALL contain a Link to a List of Neighbor instances.
- 7668 **NotificationListLink Object** (ListLink)
7669 SHALL contain a Link to a List of Notification instances.
- 7670 **PowerStatusLink Object** (Link)
7671 SHALL contain a Link to an instance of PowerStatus.
- 7672 **PrepaymentLink Object** (Link)
7673 SHALL contain a Link to an instance of Prepayment.
- 7674 **PrepaymentListLink Object** (ListLink)
7675 SHALL contain a Link to a List of Prepayment instances.
- 7676 **PrepayOperationStatusLink Object** (Link)
7677 SHALL contain a Link to an instance of PrepayOperationStatus.
- 7678 **PriceResponseCfgListLink Object** (ListLink)
7679 SHALL contain a Link to a List of PriceResponseCfg instances.
- 7680 **ProjectionReadingListLink Object** (ListLink)
7681 SHALL contain a Link to a List of ProjectionReading instances.
- 7682 **RateComponentLink Object** (Link)
7683 SHALL contain a Link to an instance of RateComponent.
- 7684 **RateComponentListLink Object** (ListLink)
7685 SHALL contain a Link to a List of RateComponent instances.

- 7686 **ReadingLink Object** (Link)
7687 A Link to a Reading.
- 7688 **ReadingListLink Object** (ListLink)
7689 SHALL contain a Link to a List of Reading instances.
- 7690 **ReadingSetListLink Object** (ListLink)
7691 SHALL contain a Link to a List of ReadingSet instances.
- 7692 **ReadingTypeLink Object** (Link)
7693 SHALL contain a Link to an instance of ReadingType.
- 7694 **RegistrationLink Object** (Link)
7695 SHALL contain a Link to an instance of Registration.
- 7696 **ResponseListLink Object** (ListLink)
7697 SHALL contain a Link to a List of Response instances.
- 7698 **ResponseSetListLink Object** (ListLink)
7699 SHALL contain a Link to a List of ResponseSet instances.
- 7700 **RPLInstanceListLink Object** (ListLink)
7701 SHALL contain a Link to a List of RPLInterface instances.
- 7702 **RPLSourceRoutesListLink Object** (ListLink)
7703 SHALL contain a Link to a List of RPLSourceRoutes instances.
- 7704 **SelfDeviceLink Object** (Link)
7705 SHALL contain a Link to an instance of SelfDevice.
- 7706 **ServiceSupplierLink Object** (Link)
7707 SHALL contain a Link to an instance of ServiceSupplier.
- 7708 **SubscriptionListLink Object** (ListLink)
7709 SHALL contain a Link to a List of Subscription instances.
- 7710 **SupplyInterruptionOverrideListLink Object** (ListLink)
7711 SHALL contain a Link to a List of SupplyInterruptionOverride instances.
- 7712 **SupportedLocaleListLink Object** (ListLink)
7713 SHALL contain a Link to a List of SupportedLocale instances.
- 7714 **TargetReadingListLink Object** (ListLink)
7715 SHALL contain a Link to a List of TargetReading instances.
- 7716 **TariffProfileLink Object** (Link)
7717 SHALL contain a Link to an instance of TariffProfile.
- 7718 **TariffProfileListLink Object** (ListLink)
7719 SHALL contain a Link to a List of TariffProfile instances.
- 7720 **TextMessageListLink Object** (ListLink)
7721 SHALL contain a Link to a List of TextMessage instances.
- 7722 **TimeLink Object** (Link)
7723 SHALL contain a Link to an instance of Time.

7724 **TimeTariffIntervalListLink Object** (ListLink)
7725 SHALL contain a Link to a List of TimeTariffInterval instances.

7726 **UsagePointLink Object** (Link)
7727 SHALL contain a Link to an instance of UsagePoint.

7728 **UsagePointListLink Object** (ListLink)
7729 SHALL contain a Link to a List of UsagePoint instances.

7730

7731

7732

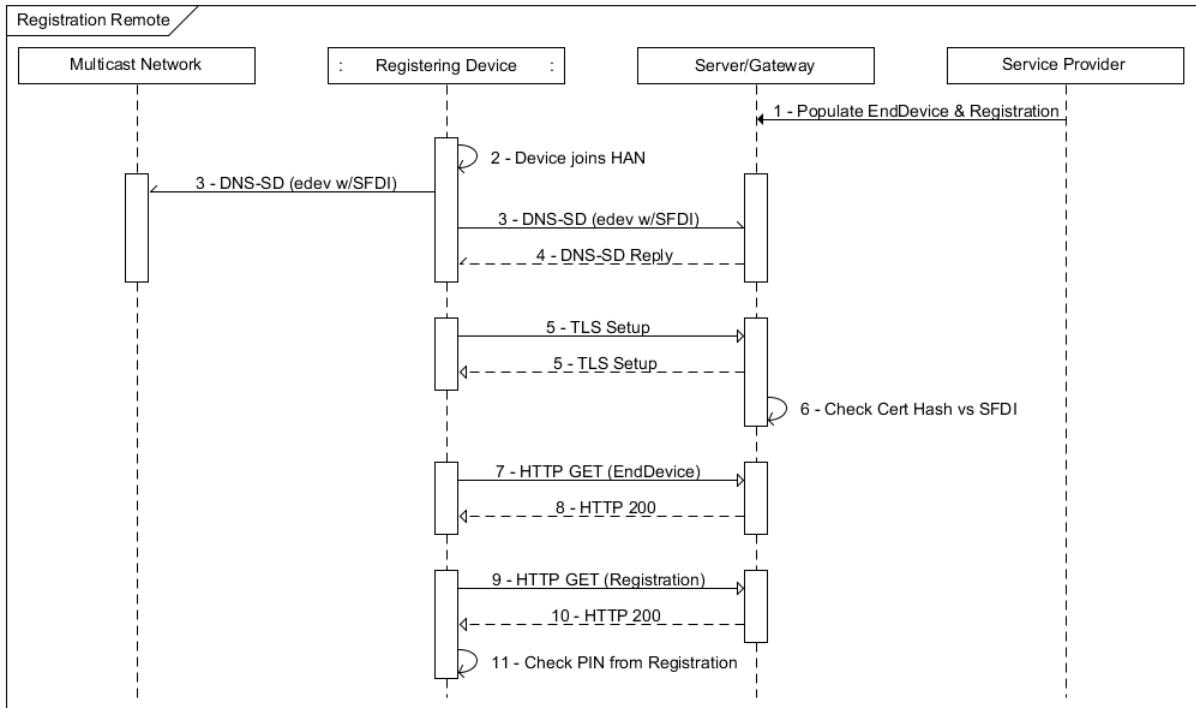
7733

7734 16 Appendix C – Examples and Guidelines [INFORMATIVE]

7735 The contents of this section are not considered to be normative and are provided for reference. Text
7736 contained in { } in the following examples are considered to be placeholders for the actual text or value.

7737 16.1 Registration: Remote

7738 This flow diagram describes client device remote registration to the customer HAN. Note that these are
7739 application layer details only, agnostic to the various layer 2 network joining techniques used by various
7740 PHY/MAC.



7741
7742 **Figure 16-1: Remote Registration**

7743 Note: An XML example in POX encoding is provided along with the EXI equivalent.

7744 **Table 16-1 POX Example: Registration Remote.**

Step	Description
1	(Out of band) Service provider populates client device's EndDevice (containing SFDI) and Registration (containing PIN) resources to appropriate HAN registration server (typically the ESI).
2	Client device joins the HAN (layer 2).
3	Client issues DNS-SD request to locate its EndDevice (keyed by its SFDI).
4	A Server or multiple servers provides DNS-SD responses with URL to client's EndDevice. If no reply is found then there are no specific registrations for this device on this network.

Step	Description
5	For each reply received the client performs TLS and client authentication and executes the following steps. Note that client certificate is sent in the clear and thus SFDI can be determined.
6	Client and server now have an encrypted connection. Each has determined it is talking to an authenticated SEP 2 device, but have not confirmed they are talking to the correct specific SEP 2 device.
7	Server verifies client identity because client certificate hashes to the client SFDI.
8	<p>Client GETs its EndDevice from Server as returned in DNS-SD Discovery</p> <p>Client sends the following request:</p> <pre>GET /edev/3 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
9	<p>Server responds with the EndDevice resource.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <EndDevice href="/edev/3" subscribable="0" xmlns="http://zigbee.org/sep"> <ConfigurationLink href="/edev/3/cfg"/> <DeviceInformationLink href="/edev/3/di"/> <DeviceStatusLink href="/edev/3/ds"/> <FileStatusLink href="/edev/3/fs"/> <PowerStatusLink href="/edev/3/ps"/> <sFDI>987654321005</sFDI> <FunctionSetAssignmentsListLink all="3" href="/edev/3/fsal"/> <RegistrationLink href="/edev/3/reg"/> <SubscriptionListLink all="0" href="/edev/3/subl"/> </EndDevice></pre>
10	<p>Client GETs its Registration (containing its PIN) from the Server as found in the EndDevice Resource.</p> <p>Client sends the following request:</p> <pre>GET /edev/3/reg HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
11	<p>Server responds with the Registration resource. Note: the PIN is thus transmitted over a secure channel.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <Registration href="/edev/3/reg" xmlns="http://zigbee.org/sep"> <pIN>123455</pIN> </Registration></pre>
12	Client verifies its PIN versus that provided by the Server. If the PIN is found to be invalid then the client knows it is not registered with this server.

7746

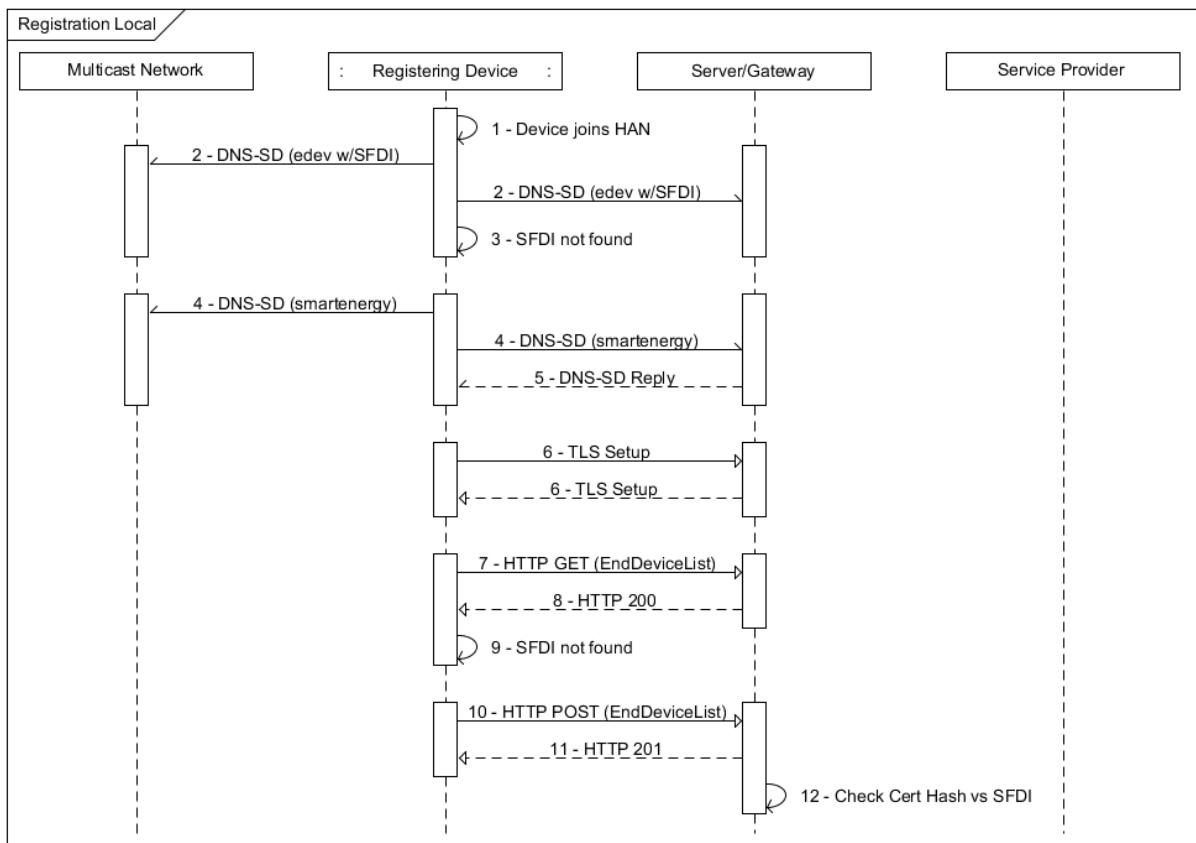
Table 16-2 EXI Example: Registration Remote.

Step	Description
1	(same as POX example)
2	(same as POX example)
3	(same as POX example)
4	<p>A Server or multiple servers provides DNS-SD responses with URL to client's EndDevice. If no reply is found then there are no specific registrations for this device on this network.</p> <p>TXT Record: level=+S0</p> <p>Note: level=+S0 indicates the server can send/receive EXI with SEP 2.0 schema with arbitrary extensions, so subsequent HTTP content on the server may contain extended parts (not used in SEP 2.0 but may be defined in future SEP 2.0.x or SEP 2.x).</p>
5	(same as POX example)
6	(same as POX example)
7	<p>Client GETs its EndDevice from Server as returned in DNS-SD Discovery.</p> <p>Client sends the following request:</p> <pre>GET /eudev/3 HTTP/1.1 Host: {hostname} Accept: application/sep-exi; level=-S0</pre> <p>Note: level=-S0 indicates the client can receive EXI with SEP 2.0 schema without any arbitrary elements and attributes.</p>
8	<p>Server responds with the EndDevice resource.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep-exi Content-Length: {contentLength}</pre> <pre>a030 114c c26a 0049 7b2b 232b b179 9800 0034 bd95 9195 d8bc ccdb 8d99 9c80 c2f6 5646 5762 f332 f646 9003 0bd9 5919 5d8b cccb d91c c018 5eca c8ca ec5e 665e cce6 80c2 f656 4657 62f3 32f7 0731 4995 07ef de04 4030 717b 2b23 2bb1 7999 7b33 9b0b 6006 97b2 b232 bb17 9997 b932 b380 0038 bd95 9195 d8bc ccdb cdd5 89b0</pre> <p>(124-bytes in binary, shown in hexadecimal)</p>
9	<p>Client GETs its Registration (containing its PIN) from the Server as found in the EndDevice Resource.</p> <p>Client sends the following request:</p> <pre>GET /eudev/3/reg HTTP/1.1 Host: {hostname} Accept: application/sep-exi; level=-S0</pre>

Step	Description
10	<p>Server responds with the Registration resource. Note: the PIN is thus transmitted over a secure channel.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep-exi Content-Length: {contentLength} a030 114c c2ef 034b d959 195d 8bcc cbdc 9959 cb96 00 (21-bytes in binary, shown in hexadecimal)</pre>
11	(same as POX example)

7747 16.2 Registration: Local

7748 This flow diagram describes client device local registration to the customer HAN. Note that these are
 7749 application layer details only, agnostic to the various layer 2 network joining techniques used by various
 7750 PHY/MAC.



7751

7752

Figure 16-2: Registration Local

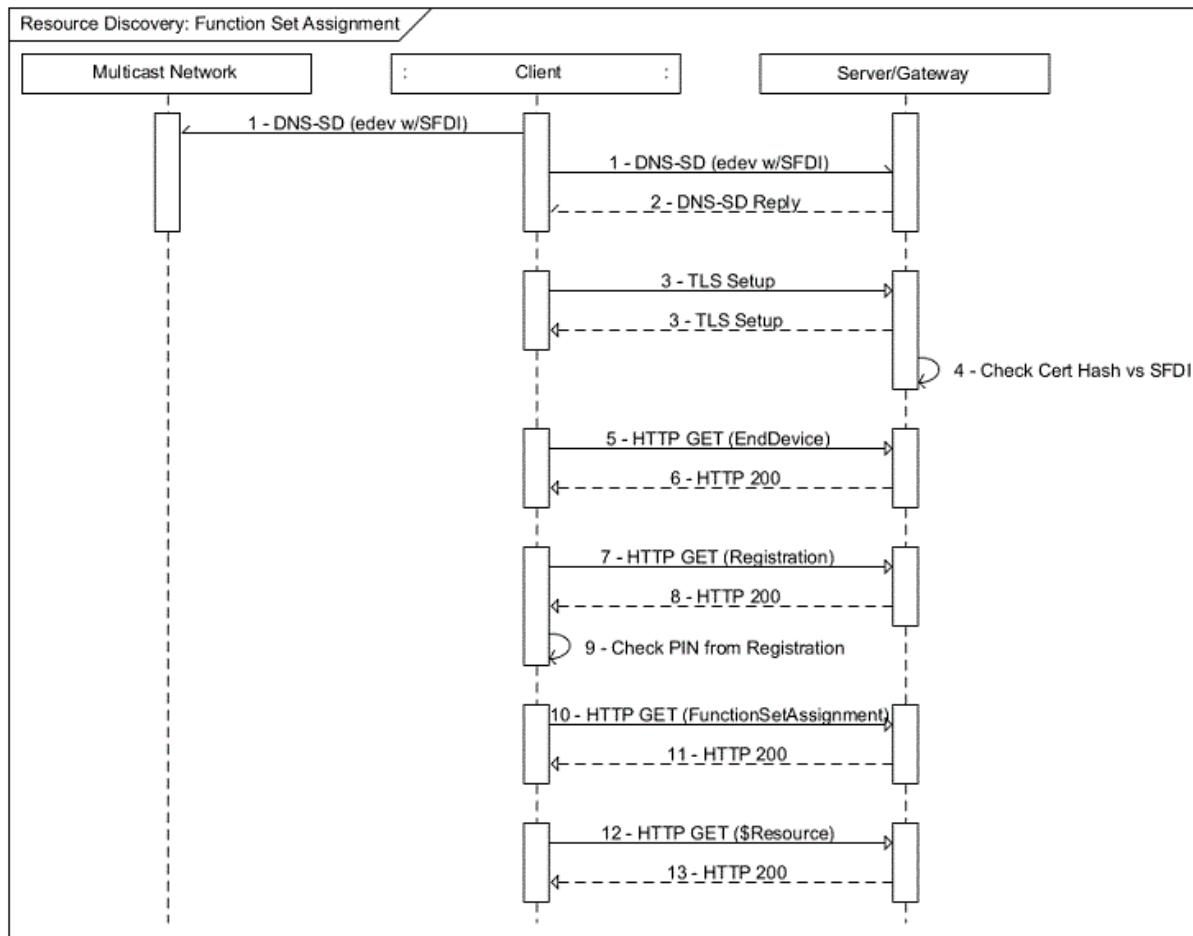
7753

Table 16-3 POX Example: Registration Local.

Step	Description
1	Client device joins the HAN (layer 2).
2	Client issues DNS-SD request to locate its EndDevice (keyed by its SFDI).
3	Client does not find a desired SFDI response.
4	Client issues DNS-SD request to locate any 'smartenergy' server.
5	A Server or multiple servers provides DNS-SD responses. If no reply is found then no 'smartenergy' servers are present on the network.
6	For each reply received the client performs TLS and client authentication and executes the following steps. Note that client certificate is sent in the clear and thus SFDI can be determined. Client and server now have an encrypted connection. Each has determined it is talking to an authenticated SEP 2 device, but have not confirmed they are talking to the correct specific SEP 2 device.
7	Client GETs the EndDeviceList from Server as returned in DNS-SD Discovery. Client sends the following request: GET /edev HTTP/1.1 Host: {hostname} Accept: application/sep+xml
8	Server responds with the EndDeviceList resource. Server sends the following response: HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <EndDeviceList all="1" href="/edev" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <EndDevice href="/edev/3" subscribable="0"> <ConfigurationLink href="/edev/3/cfg"/> <DeviceInformationLink href="/edev/3/di"/> <DeviceStatusLink href="/edev/3/ds"/> <FileStatusLink href="/edev/3/fs"/> <PowerStatusLink href="/edev/3/ps"/> <sFDI>987654321005</sFDI> <FunctionSetAssignmentsListLink all="3" href="/edev/3/fsal"/> <RegistrationLink href="/edev/3/reg"/> <SubscriptionListLink all="0" href="/edev/3/sub1"/> </EndDevice> </EndDeviceList>
9	Client does not find its SFDI in any of the EndDevices in the EndDeviceList.

Step	Description
10	<p>Client does a POST to EndDeviceList to add its EndDevice to the server.</p> <p>Client sends the following request:</p> <pre>POST /eudev HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <EndDevice href="/eudev/3" xmlns="http://zigbee.org/sep"> <sFDI>987654321005</sFDI> </EndDevice></pre>
11	<p>Server indicates the EndDevice Record was created with the following:</p> <p>If new EndDevice entry is created, the Server would respond with the following where Location indicates the path to the newly created EndDevice resource:</p> <pre>HTTP/1.1 201 Created Location: /eudev/4</pre>
12	Server verifies client identity because client certificate hashes to the client SFDI.

7754 16.3 Discovery: Function Set Assignment



7755

7756

Figure 16-3: Discovery: Function Set Assignment

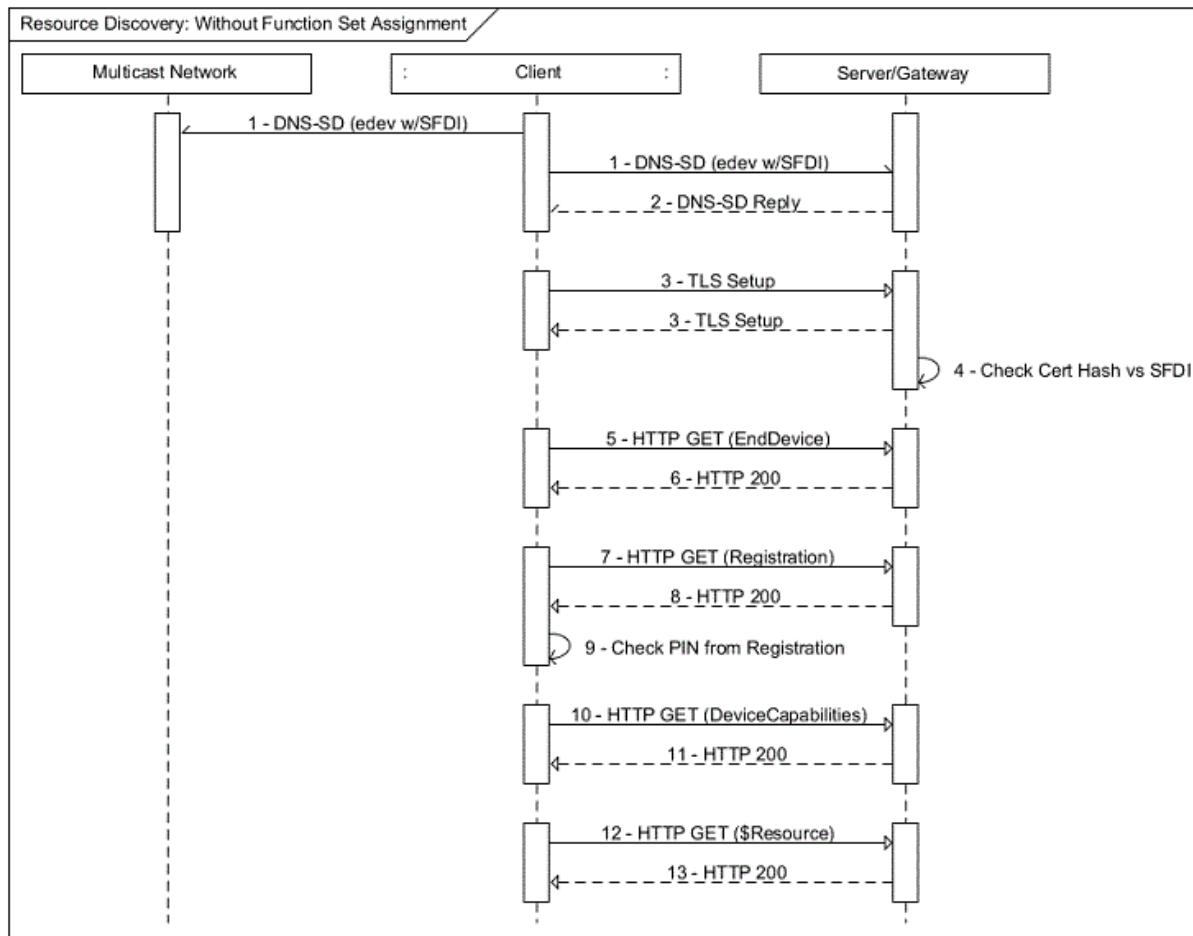
7757 Note: The Client finds a Server with a validated *EndDevice* record. The *EndDevice* Record is found to
 7758 have a *FunctionSetAssignmentListLink* with a *FunctionSetAssignment* record that contains the desired
 7759 Resource.

7760 **Table 16-4 POX Example: Discovery Function Set Assignment**

Step	Description
1	(See Registration Examples)
2	(See Registration Examples)
3	(See Registration Examples)
4	(See Registration Examples)
5	(See Registration Examples)

Step	Description
6	<p>Server responds with the EndDevice resource with a FunctionSetAssignmentListLink.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <EndDevice href="/edev/3" subscribable="0" xmlns="http://zigbee.org/sep"> <ConfigurationLink href="/edev/3/cfg"/> <DeviceInformationLink href="/edev/3/di"/> <DeviceStatusLink href="/edev/3/ds"/> <FileStatusLink href="/edev/3/fs"/> <PowerStatusLink href="/edev/3/ps"/> <sFDI>987654321005</sFDI> <FunctionSetAssignmentsListLink all="2" href="/edev/3/fsal"/> <RegistrationLink href="/edev/3/reg"/> <SubscriptionListLink all="0" href="/edev/3/subl"/> </EndDevice></pre>
7	(See Registration Examples)
8	(See Registration Examples)
9	(See Registration Examples)
10	<p>Client GETs its FunctionSetAssignment from the Server as found in the EndDevice Resource.</p> <p>Client sends the following request:</p> <pre>GET /edev/3/fsal?l=2 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
11	<p>Server responds with the FunctionSetAssignment resource.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <FunctionSetAssignmentsList all="2" href="/edev/3/fsal" results="2" subscribable="0" xmlns="http://zigbee.org/sep"> <FunctionSetAssignments href="/edev/3/fsal/0" subscribable="0"> <DemandResponseProgramListLink all="2" href="/edev/3/fsal/0/drpl"/> <MessagingProgramListLink all="1" href="/edev/3/fsal/0/msg1"/> <mRID>0ED30F5A0000</mRID> <description>FunctionSetAssignment Suave</description> </FunctionSetAssignments> <FunctionSetAssignments href="/edev/3/fsal/1" subscribable="0"> <MessagingProgramListLink all="1" href="/edev/3/fsal/1/msg1"/> <mRID>0ED30F5A0001</mRID> <description>FunctionSetAssignment Guttural</description> </FunctionSetAssignments> </FunctionSetAssignmentsList></pre>
12	<p>Client GETs its desired Resource from the Server as found in the FunctionSetAssignment Resource. In this case, the Resource of choice is a DemandResponseProgram.</p> <p>(see DemandResponse Examples)</p>
13	(see DemandResponse Examples)

7761 16.4 **Discovery: Without Function Set Assignment**



7762

7763

Figure 16-4: Discovery Without Function Set Assignment

7764 Note: The Client finds a Server with a validated *EndDevice* record. The *EndDevice* Record is not found to have a *FunctionSetAssignmentListLink* with a *FunctionSetAssignment* record that contains the desired Resource.

7765

7766

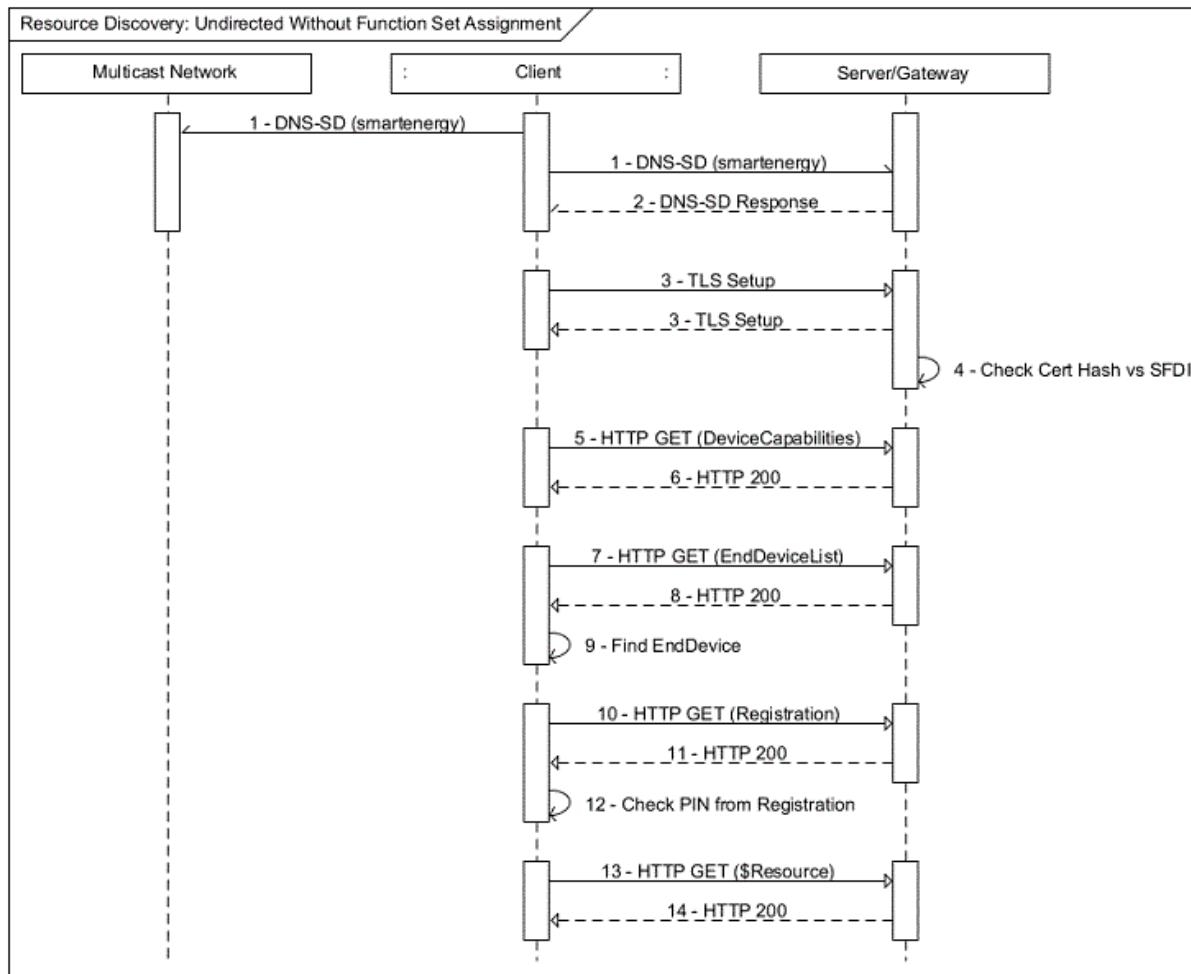
7767

Table 16-5 POX Example: Discovery Without Function Set Assignment.

Step	Description
1	(See Registration Examples)
2	(See Registration Examples)
3	(See Registration Examples)
4	(See Registration Examples)
5	(See Registration Examples)

Step	Description
6	<p>Server responds with the EndDevice resource without a FunctionSetAssignmentListLink. In this case, the Resource of choice is a DemandResponseProgram which is not a direct member of EndDevice.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <EndDevice href="/edev/3" subscribable="0" xmlns="http://zigbee.org/sep"> <ConfigurationLink href="/edev/3/cfg"/> <DeviceInformationLink href="/edev/3/di"/> <DeviceStatusLink href="/edev/3/ds"/> <FileStatusLink href="/edev/3/fs"/> <PowerStatusLink href="/edev/3/ps"/> <sFDI>987654321005</sFDI> <RegistrationLink href="/edev/3/reg"/> <SubscriptionListLink all="0" href="/edev/3/subl"/> </EndDevice></pre>
7	(See Registration Examples)
8	(See Registration Examples)
9	(See Registration Examples)
10	<p>Client GETs its DeviceCapabilities from the Server as found in the DNS-SD TXT Record.</p> <p>Client sends the following request:</p> <pre>GET /dcap HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
11	<p>Server responds with the DeviceCapabilities resource.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <DeviceCapability href="/dcap" xmlns="http://zigbee.org/sep"> <DemandResponseProgramListLink all="1" href="/drp"/> <MessagingProgramListLink all="2" href="/msg"/> <EndDeviceListLink all="1" href="/edev"/> <SelfDeviceLink href="/sdev"/> </DeviceCapability></pre>
12	<p>Client GETs its desired Resource from the Server as found in the DeviceCapabilities Resource. In this case, the Resource of choice is a DemandResponseProgram.</p> <p>(see DemandResponse Examples)</p>
13	(see DemandResponse Examples)

7768 16.5 **Discovery: Undirected Without Function Set Assignment**



7769

7770

Figure 16-5: Discovery Undirected Without Function Set Assignment

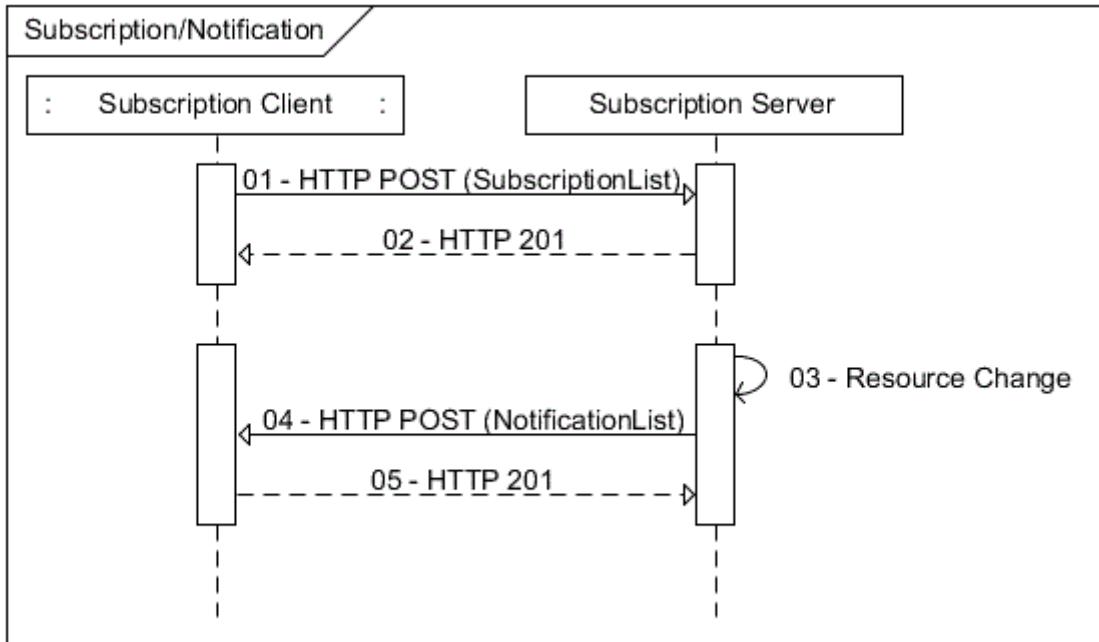
7771 Note: The Client discovers all Servers and searches through the records to find a Server with a validated
 7772 *EndDevice* record for itself. The *EndDevice* Record is not found to have a *FunctionSetAssignmentListLink*
 7773 with a *FunctionSetAssignment* record that contains the desired Resource. The Client searches for the
 7774 desired Resource in *EndDevice* or in *DeviceCapabilities*.

7775 **Table 16-6 POX Example: Discovery Undirected Without Function Set Assignment.**

Step	Description
1	(See Registration: Local Example)
2	(See Registration: Local Example)
3	(See Registration: Local Example)
4	(See Registration: Local Example)

Step	Description
5	(See Resource Discovery: Without Function Set Assignment Example)
6	(See Resource Discovery: Without Function Set Assignment Example)
7	(See Registration: Local Example)
8	(See Registration: Local Example)
9	The Client searches through the List of EndDevice Records to find one with a matching SFDI. If not found, the Server contains no records for the Client.
10	(See Registration: Remote Example)
11	(See Registration: Remote Example)
12	(See Registration: Remote Example)
13	(See Resource Discovery: Without Function Set Assignment Example)
14	(See Resource Discovery: Without Function Set Assignment Example)

7776 16.6 **Subscription / Notification**



7777

7778

Figure 16-6: Subscription / Notification

7779 Note: An XML example in POX encoding is provided along with the EXI equivalent.

7780

Table 16-7 POX Example: Subscription / Notification.

Step	Description
1	<p>After discovering the URI of the server's SubscriptionList and desired Resource, the Client can append new subscription entries to the list.</p> <p>Note: In this case, the Client is conditionally subscribing to an instantaneous meter reading value. If the value is less than 10 or greater than 1000, a notification is sent to the specified notificationURI in the POX encoded format.</p> <pre>POST /edev/8/sub HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Subscription xmlns="http://zigbee.org/sep"> <subscribedResource>/upt/0/mr/4/r</subscribedResource> <Condition> <attributeIdentifier>0</attributeIdentifier> <lowerThreshold>10</lowerThreshold> <upperThreshold>1000</upperThreshold> </Condition> <encoding>0</encoding> <level>+S0</level> <limit>1</limit> <notificationURI>/note</notificationURI> </Subscription></pre>
2	<p>If new subscription entries are created, the Server would respond with the following, where Location indicates the path to the newly created Subscription resource:</p> <pre>HTTP/1.1 201 Created Location: /edev/8/sub/5</pre> <p>If entries were not created but modified the Server would respond with:</p> <pre>HTTP/1.1 204 No Content</pre>
3	A change on the subscribed resource occurs which satisfies the above specified Conditions.
4	<p>Notification is sent from the Server to the Client:</p> <pre>POST /note HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Notification xmlns="http://zigbee.org/sep"> <subscribedResource>/upt/0/mr/4/r</subscribedResource> <Resource xsi:type="Reading"> <timePeriod> <duration>0</duration> <start>12987364</start> </timePeriod> <value>1001</value> </Resource> <status>0</status> <subscriptionURI>/edev/8/sub/5</subscriptionURI> </Notification></pre>
5	The Client responds with HTTP Response: <pre>HTTP/1.1 201 Created</pre>

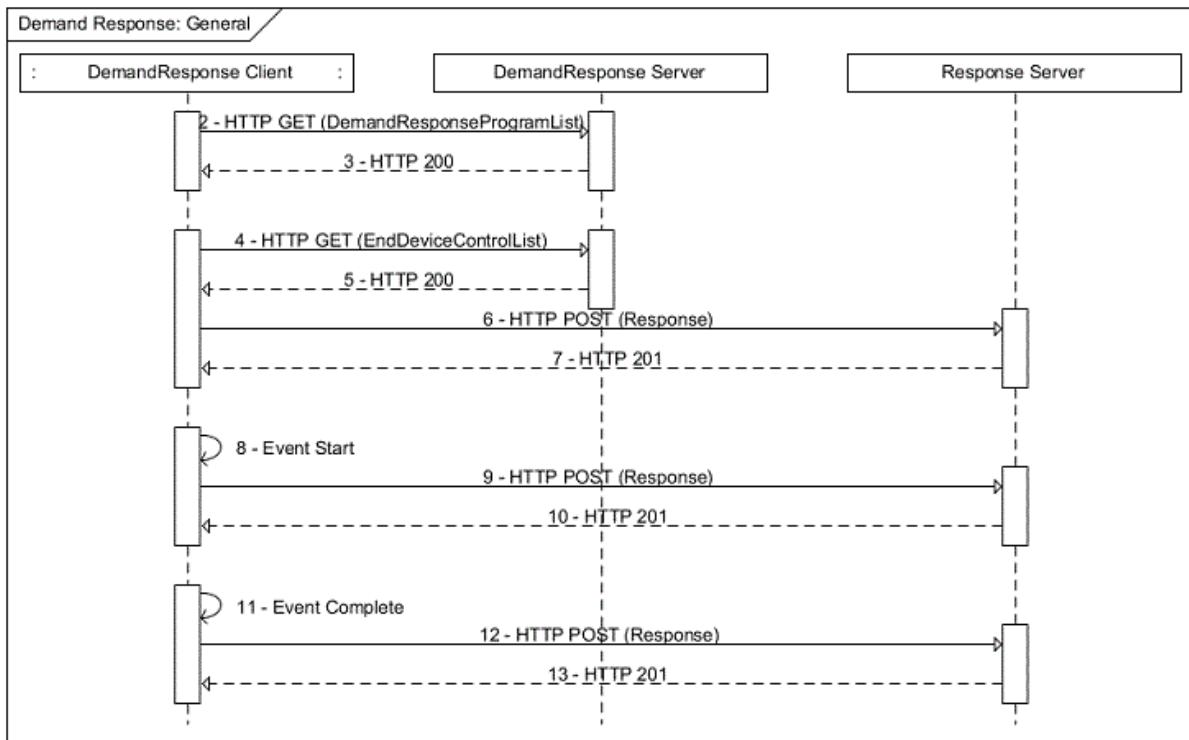
7781

7782

Table 16-8 EXI Example: Subscription / Notification.

Step	Description
1	(same as POX example with the following exceptions) POST format is set with <i>Content-Type: application/sep-exi</i> Notification encoding type is specified with: <i><encoding>1</encoding></i> Notification schema version and options are specified with <i><level>-S0</level></i> POST /eudev/8/sub HTTP/1.1 Host: {hostname} Content-Type: application/sep-exi Content-Length: {contentLength} a030 114c c30c 41e5 eea5 0e85 e605 edae 45e6 85ee 4000 00a0 e807 0010 0a5a a660 0040 397b 737b a328 (40-bytes in binary, shown in hexadecimal)
2	(same as POX example)
3	(same as POX example)
4	(same as POX example with the following exceptions) POST format is set with <i>Content-Type: application/sep-exi</i> POST /note HTTP/1.1 Host: {hostname} Content-Type: application/sep-exi Content-Length: {contentLength} a030 114c c2b5 41e5 eea5 0e85 e605 edae 45e6 85ee 4614 0100 63a4 1c80 003c bd95 9195 d8bc e0bd cdd5 88bc d4 (43-bytes in binary, shown in hexadecimal)
5	(same as POX example)

7783 16.7 Demand Response: General



7784

7785

Figure 16-7: Demand Response General

7786

Table 16-9 POX Example: Demand Response – General.

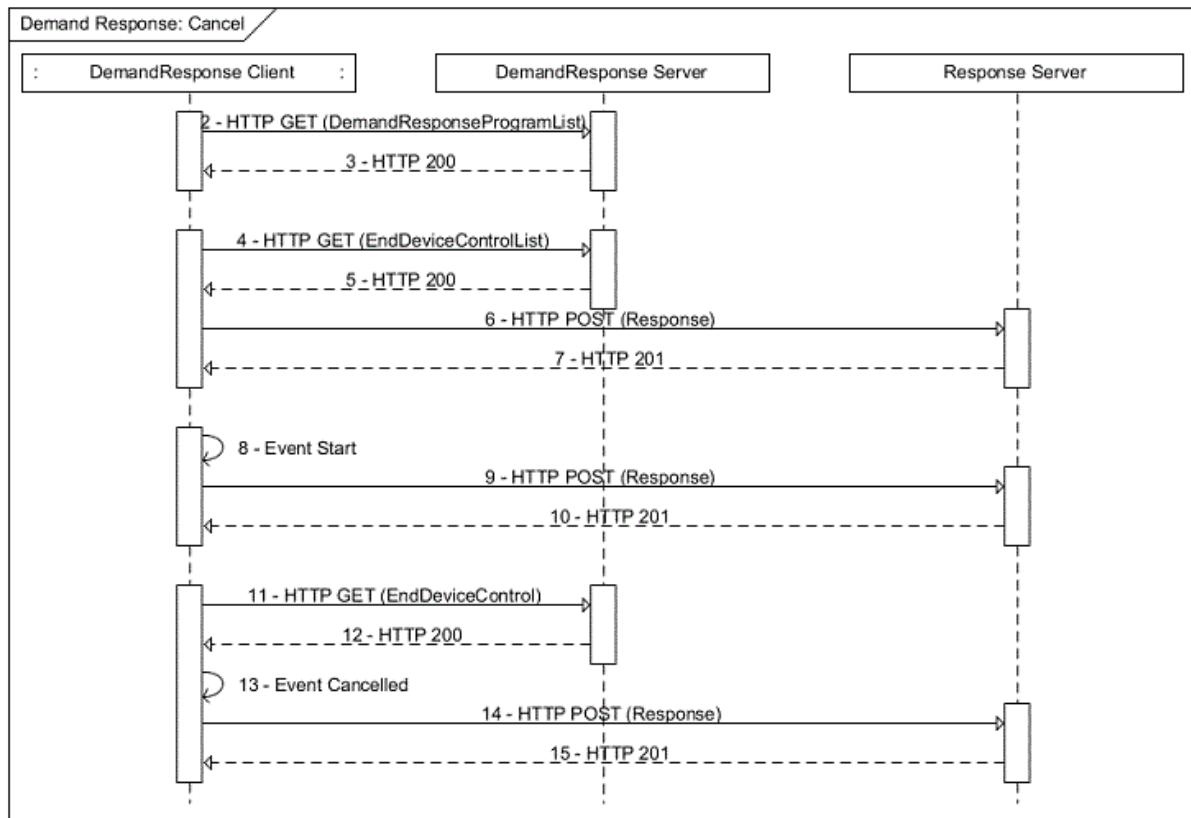
Step	Description
1	The client has discovered its EndDevice instance on the Server with a link to its FunctionSetAssignments. Within its FunctionSetAssignments, the client discovers it is part of a DemandResponseProgram. The enrollment is provided out of band.
2	Client GETs the list of DemandResponsePrograms from the DRLC Server. Client sends the following request: <pre>GET /drp?l=2 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
3	<p>DemandResponse server responds with the list of DemandResponsePrograms.</p> <p>Server sends the following response:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <DemandResponseProgramList all="2" results="2" subscribable="0" xmlns="http://zigbee.org/sep"> <DemandResponseProgram href="/drp/1"> <mRID>0FB7</mRID> <description>Operation X</description> <ActiveEndDeviceControlListLink all="0" href="/drp/1/aedc"/> <EndDeviceControlListLink all="1" href="/drp/1/edc"/> <primacy>0</primacy> </DemandResponseProgram> <DemandResponseProgram href="/drp/2"> <mRID>80000001</mRID> <description>The Wackness</description> <ActiveEndDeviceControlListLink all="0" href="/drp/2/aedc"/> <EndDeviceControlListLink all="0" href="/drp/2/edc"/> <primacy>1</primacy> </DemandResponseProgram> </DemandResponseProgramList></pre>
4	<p>Client GETs the list of EndDeviceControls from the DRLC Server for the desired DemandResponseProgram.</p> <p>Client sends the following request:</p> <pre> GET /drp/1/edc HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
5	<p>DemandResponse server responds with the list of EndDeviceControls.</p> <p>Server sends the following response:</p> <pre data-bbox="274 386 1237 1136"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <EndDeviceControlList all="1" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <EndDeviceControl href="/drp/1/edc" replyTo="{hostname}/rsp" responseRequired="01" subscribable="0"> <mRID>CAFEFEED</mRID> <description>Emergency One Hour Coffee Brew</description> <creationTime>1234556</creationTime> <EventStatus> <currentStatus>0</currentStatus> <dateTime>1234556</dateTime> <potentiallySuperseded>false</potentiallySuperseded> <reason>Need Caffeine Soon</reason> </EventStatus> <interval> <duration>360</duration> <start>1234900</start> </interval> <randomizeDuration>60</randomizeDuration> <randomizeStart>60</randomizeStart> <deviceCategory>08</deviceCategory> <drProgramMandatory>true</drProgramMandatory> <loadShiftForward>true</loadShiftForward> <SetPoint> <heatingSetpoint>10000</heatingSetpoint> </SetPoint> </EndDeviceControl> </EndDeviceControlList></pre>
6	<p>For each EndDeviceControl with ResponseRequired Bit 0, set the Client POSTs a Response with a Status of "Event Received" to the Response resource specified by the replyTo field in the EndDeviceControl.</p> <p>Client sends the following:</p> <pre data-bbox="274 1294 915 1558"> POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Response xmlns="http://zigbee.org/sep"> <createdDateTime>1234560</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>1</status> <subject>CAFEFEED</subject> </Response></pre>
7	<p>Response Server replies with Status 201 Created:</p> <pre data-bbox="274 1622 535 1643">HTTP/1.1 201 Created</pre>
8	<p>Client begins the event at the defined start Time.</p>

Step	Description
9	<p>Client POST a Response with a Status of "Event Started" to the Response resource specified by the replyTo field in the EndDeviceControl.</p> <p>Client sends the following:</p> <pre>POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Response xmlns="http://zigbee.org/sep"> <createdDateTime>123490</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>2</status> <subject>CAFEFEED</subject> </Response></pre>
10	<p>Response Server Responds with Status 201 Created:</p> <p>HTTP/1.1 201 Created</p>
11	<p>Client completes the event.</p>
12	<p>Client POST a Response with a Status of "Event Completed" to the Response resource specified by the replyTo field in the EndDeviceControl.</p> <p>Client sends the following:</p> <pre>POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Response xmlns="http://zigbee.org/sep"> <createdDateTime>1235260</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>3</status> <subject>CAFEFEED</subject> </Response></pre>
13	<p>Response Server Responds with Status 201 Created:</p> <p>HTTP/1.1 201 Created</p>

7787 16.8 Demand Response: Cancel



7788

7789

Figure 16-8: Demand Response - Cancel

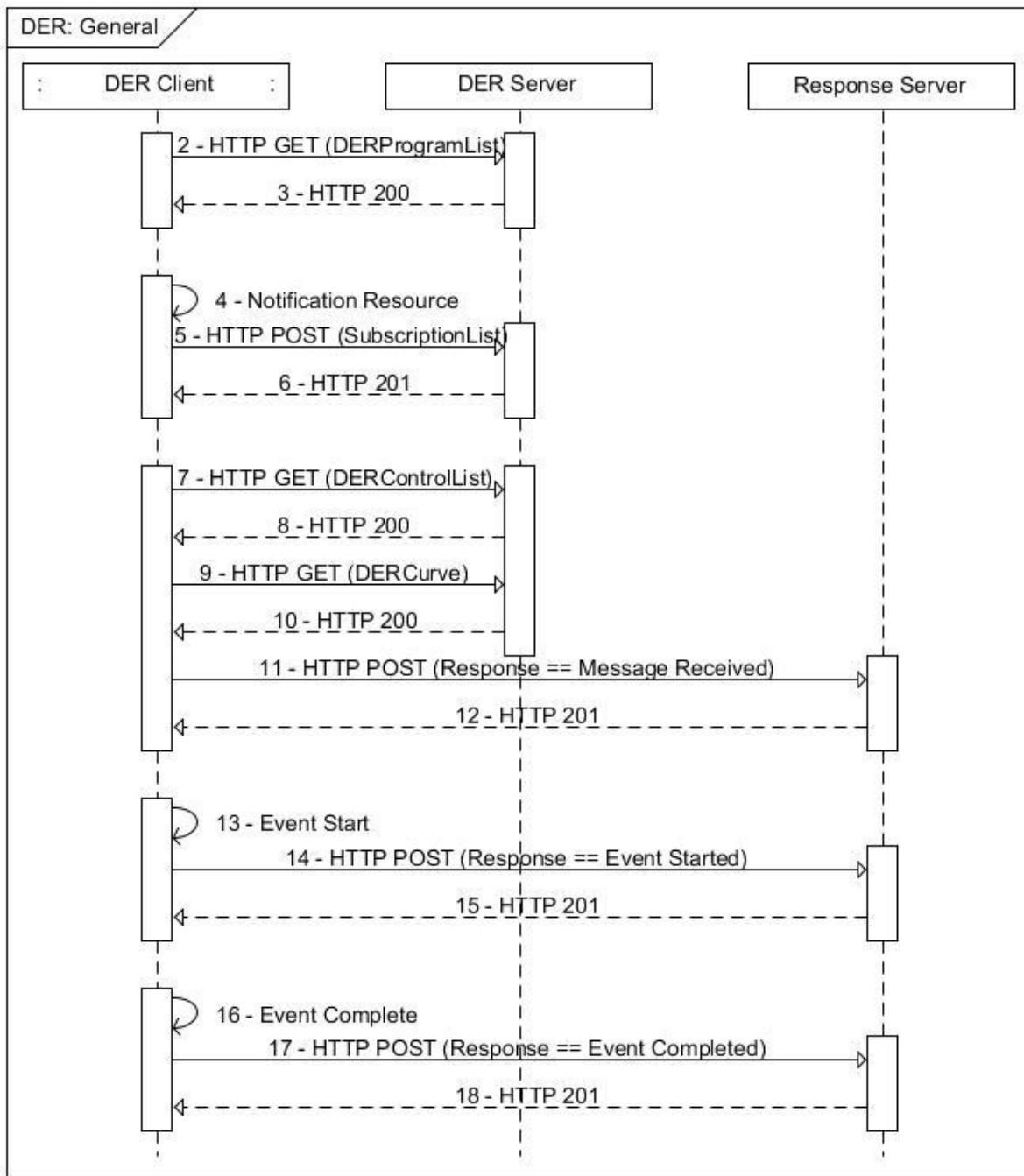
7790

Table 16-10 POX Example: Demand Response - Cancel.

Step	Description
1	(Same as Demand Response: General)
2	(Same as Demand Response: General)
3	(Same as Demand Response: General)
4	(Same as Demand Response: General)
5	(Same as Demand Response: General)
6	(Same as Demand Response: General)
7	(Same as Demand Response: General)
8	(Same as Demand Response: General)
9	(Same as Demand Response: General)

Step	Description
10	(Same as Demand Response: General)
11	<p>Client GETs a specific EndDeviceControl from the DRLC Server to check on the current Status of the event it's executing. This should happen on a periodic basis to check if the control is cancelled.</p> <p>Client sends the following request:</p> <pre>GET /drp/1/edc/3 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
12	<p>DemandResponse Server indicates the event is cancelled:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <EndDeviceControl href="/drp/1/edc/3" replyTo="{hostname}/rsp" responseRequired="01" subscribable="0" xmlns="http://zigbee.org/sep"> <mRID>CAFEFEED</mRID> <description>Emergency One Hour Coffee Brew</description> <creationTime>1234556</creationTime> <EventStatus> <currentStatus>2</currentStatus> <dateTime>1234960</dateTime> <potentiallySuperseded>false</potentiallySuperseded> <reason>Caffeine Overload</reason> </EventStatus> <interval> <duration>360</duration> <start>1234900</start> </interval> <randomizeDuration>60</randomizeDuration> <randomizeStart>60</randomizeStart> <deviceCategory>0008</deviceCategory> <drProgramMandatory>true</drProgramMandatory> <loadShiftForward>true</loadShiftForward> <SetPoint> <heatingSetpoint>10000</heatingSetpoint> </SetPoint> </EndDeviceControl></pre>
13	Client cancels the event.
14	<p>Client responds with the specific instance of the EndDeviceControl with an updated EventStatus of 'Cancelled'</p> <p>Server sends the following response:</p> <pre>POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Response xmlns="http://zigbee.org/sep"> <createdDateTime>1235100</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>6</status> <subject>CAFEFEED</subject> </Response></pre>
15	Response Server replies with Status 201 Created:
	<pre>HTTP/1.1 201 Created</pre>

7791 16.9 Distributed Energy Resource: General



7792

7793

Figure 16-9: Distributed Energy Resource: General

7794

Table 16-11 POX Example: Distributed Energy Resource - General.

Step	Description
1	The client discovers a server of its EndDevice instance and GETs its FunctionSetAssignments (FSA). Through its FSA the client discovers it is enrolled in a DERProgram (enrollment occurs out of band). The client registers with the specified DER program server if a secure connection is required.
2	<p>Client GETs the DERProgramList specified in its FSA. Client sends the following request, in this case indicating that it can accept either EXI or XML:</p> <pre>GET /derp HTTP/1.1 Host: {hostname} Accept: application/sep+xml; level=+S0</pre>
3	<p>The DER server responds with the requested DERProgram list, for example:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <DERProgramList xmlns="http://zigbee.org/sep" href="/derp" all="1" results="1"> <DERProgram href="/derp/0"> <mRID>01BE7A7E57</mRID> <description>Example DER Program</description> <DERControlListLink all="2" href="/derp/0/derc" /> <primacy>2</primacy> </DERProgram> </DERProgramList></pre>
4	The client locates or creates a local Notification resource that can be used to receive notification of changes to the DERControlList.
5	<p>The client POSTs the URI of the Notification resource, together with the DERControlListLink it read from the DERProgram, to the subscription list of its EndDevice instance on the DER server.</p> <pre>POST /eudev/0/sub HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Subscription xmlns="http://zigbee.org/sep"> <subscribedResource>http://server.example.com/derp/0/derc</subscribedResource> <encoding>0</encoding> <level>+S0</level> <limit>0</limit> <notificationURI>http://client.example.com/ntfy</notificationURI> </Subscription></pre>
6	The DER server responds with Status 201 Created. In the future, the DER server will "push" DERControlList updates to the client.
	<pre>HTTP/1.1 201 CREATED Location: /eudev/0/sub/1 Content-Length: 0</pre>
7	Client GETs the list of DERControls from the DER server. Client sends the following request, in this case asking for the first element of the list:
	<pre>GET /derp/0/derc?s=0&l=1 HTTP/1.1 Host: {hostname}</pre>

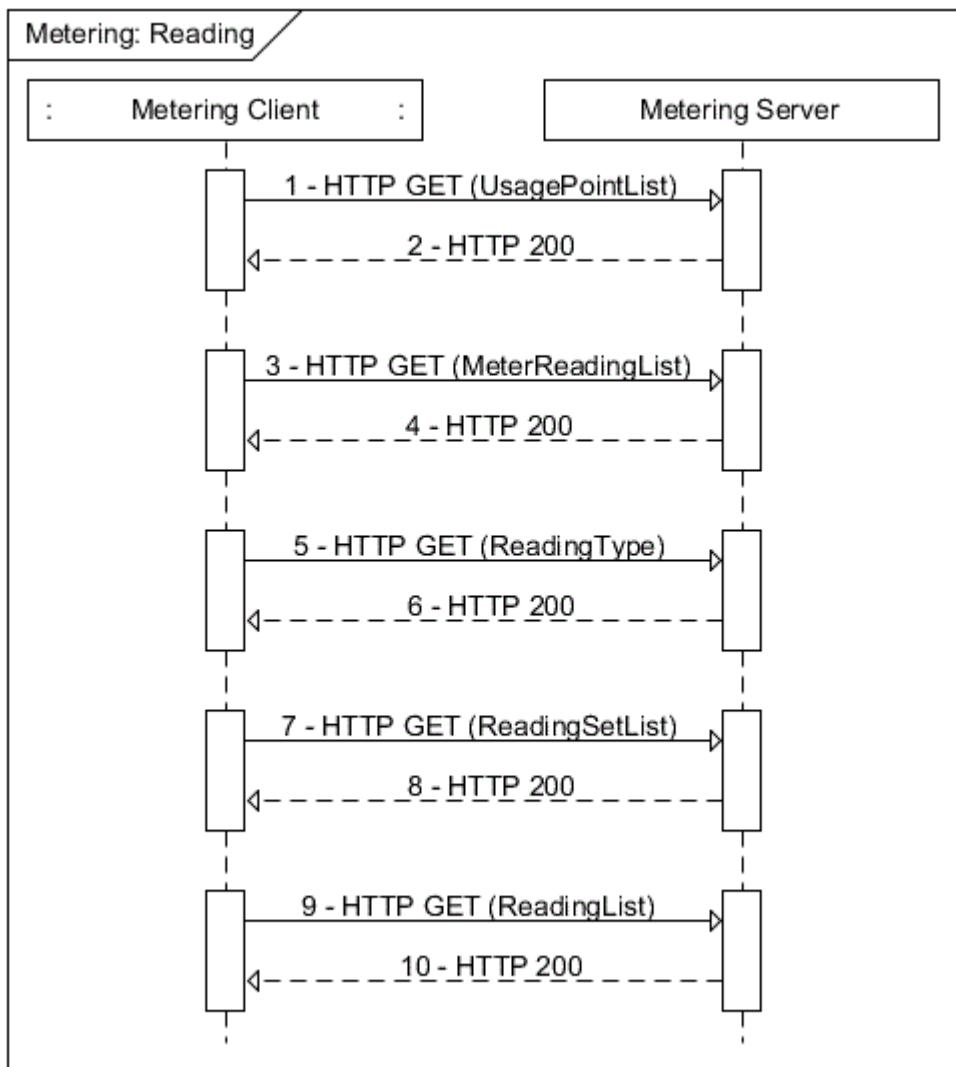
	Accept: application/sep+xml; level=+S0
8	<p>DER server responds with the first DERControl on the list. Server sends the following response:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <DERControlList xmlns="http://zigbee.org/sep" all="2" href="/derp/0/derc" results="1" subscribable="1"> <DERControl> <mRID>02BE7A7E57</mRID> <description>Example DERControl 1</description> <creationTime>1341446390</creationTime> <EventStatus> <currentStatus>1</currentStatus> <dateTime>1341532800</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <duration>86400</duration> <start>1341446400</start> </interval> <randomizeDuration>180</randomizeDuration> <randomizeStart>180</randomizeStart> <DERControlBase> <opModVoltVAr href="/derp/0/dc/3"/> </DERControlBase> </DERControl> </DERControlList></pre>
9	<p>The DERControl above calls for dynamic (curve-based) Volt-VAr control mode. The specified curve URI is "/derp/0/dc/3". The client GETs the DERCurve:</p> <pre> GET /derp/0/dc/3 HTTP/1.1 Host: {hostname} Accept: application/sep+xml; level=+S0</pre>
10	<p>The DER server responds with the requested DERCurve. Client should check the curveType to ensure it is Volt-VAr. In this example (see Figure 12-2) the delivered reactive power remains at 50% of statVArAvail (positive sign indicates delivered or over-excited, yRefType==3 indicates %statVArAvail) as long as the effective percent voltage is at or below 99% of nominal. When the voltage is at 100% of nominal, no reactive power is delivered. As the voltage climbs above nominal, reactive power is received (negative sign indicates under-excited). Voltage may jitter slightly within the dead band created by the four curve points without affecting the reactive power output.</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <DERCurve xmlns="http://zigbee.org/sep" href="/derp/0/dc/3"> <mRID>04BE7A7E57</mRID> <description>An example Volt-VAr curve</description> <creationTime>1341446380</creationTime> <CurveData> <xvalue>99</xvalue> <yvalue>50</yvalue> </CurveData> <CurveData> <xvalue>103</xvalue> <yvalue>-50</yvalue> </CurveData> <CurveData></pre>

	<pre> <xvalue>101</xvalue> <yvalue>-50</yvalue> </CurveData> <CurveData> <xvalue>97</xvalue> <yvalue>50</yvalue> </CurveData> <curveType>0</curveType> <rampDecTms>600</rampDecTms> <rampIncTms>600</rampIncTms> <rampPT1Tms>10</rampPT1Tms> <xMultiplier>0</xMultiplier> <yMultiplier>0</yMultiplier> <yRefType>3</yRefType> </DERCurve></pre>
11	<p>For each DERControl with ResponseRequired Bit 0 set, the Client POSTs a Response with a Status of "Message Received" to the Response resource specified by the replyTo field in the DERControl. Client sends the following:</p> <pre> POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Response xmlns="http://zigbee.org/sep"> <createdDateTime>1341507000</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>1</status> <subject>0002BE7A7E57</subject> </Response></pre>
12	<p>Response Server responds with:</p> <pre>HTTP/1.1 201 Created</pre>
13	<p>Client begins the event at the specified start (or current) time.</p>
14	<p>Client POSTs a Response with a Status of "Event Started" to the Response resource specified by the replyTo field in the DERControl. Client sends the following:</p> <pre> POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <Response xmlns="http://zigbee.org/sep"> <createdDateTime>1341507010</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>2</status> <subject>0002BE7A7E57</subject> </Response></pre>
15	<p>Response Server responds with:</p> <pre>HTTP/1.1 201 Created</pre>
16	<p>Client completes the event.</p>
17	<p>Client POST a Response with a Status of "Event Completed" to the Response resource specified by the replyTo field in the DERControl. Client sends the following:</p> <pre> POST /rsp HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength}</pre>

	<Response xmlns="http://zigbee.org/sep"> <createdDateTime>1341532810</createdDateTime> <endDeviceLFDI>C0FFEE00</endDeviceLFDI> <status>3</status> <subject>0002BE7A7E57</subject> </Response>
18	Response Server responds with: HTTP/1.1 201 Created

7795 16.10 **Metering: Reading**

7796 This is a use case where a metering function set client (e.g., In-Premises Display) queries a reading from a
7797 usage point. For this example, we will assume the meter is configured for 4 TOU tiers and no Blocks and
7798 that we want to read the current tier 3 consumption.



7799

7800

Figure 16-10: Meter Reading

7801

Note: In most cases, registration is required to obtain access to metering.

7802

Table 16-12 POX Example: Meter Reading.

Step	Description
1	<p>Client GETs the UsagePointList from the Metering Server</p> <p>Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.</p> <p>Client sends the following request:</p> <pre>GET /upt HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
2	<p>Metering Server replies with UsagePointList.</p> <p>A typical response looks like:</p> <pre data-bbox="298 386 1122 747"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <UsagePointList all="1" href="/upt" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <UsagePoint href="/upt/0"> <mRID>0B00006CC8</mRID> <description>Usage Point</description> <roleFlags>12</roleFlags> <serviceCategoryKind>0</serviceCategoryKind> <status>1</status> <MeterReadingListLink all="6" href="/upt/0/mr"/> </UsagePoint> </UsagePointList></pre>
3	<p>Client GETs the MeterReadingList from the Metering Server.</p> <p>Note: This and the next 3 steps may be repeated for each page required to read the entire list. For this example, we are requesting up to 10 MeterReadings at a time. Subsequent GETs would increment the "s" query parameter by 10 or however many list items are returned.</p> <p>Client sends the following request:</p> <pre data-bbox="298 967 698 1036"> GET /upt/0/mr?s=0&l=10 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
4	<p>Metering Server replies with up to 10 MeterReadingList instances. Only 6 are returned in this case as indicated by the MeterReadingListLink "all" attribute in step 2.</p> <p>A typical response looks like:</p> <pre data-bbox="298 1174 1212 1826"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <MeterReadingList all="6" href="/upt/0/mr" results="6" subscribable="0" xmlns="http://zigbee.org/sep"> <MeterReading href="/upt/0/mr/0"> <mRID>0C00006CC8</mRID> <description>Cumulative Reading for Wh</description> <ReadingSetListLink all="1" href="/upt/0/mr/0/rs"/> <ReadingTypeLink href="/upt/0/mr/0/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/1"> <mRID>0E00006CC8</mRID> <description>Cumulative Reading for VAR's</description> <ReadingSetListLink all="1" href="/upt/0/mr/1/rs"/> <ReadingTypeLink href="/upt/0/mr/1/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/2"> <mRID>1000006CC8</mRID> <description>Interval Reading for Wh</description> <ReadingSetListLink all="24" href="/upt/0/mr/2/rs"/> <ReadingTypeLink href="/upt/0/mr/2/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/3"> <mRID>1200006CC8</mRID> <description>Interval Reading for VAR's</description> <ReadingSetListLink all="24" href="/upt/0/mr/3/rs"/> </MeterReading> </MeterReadingList></pre>

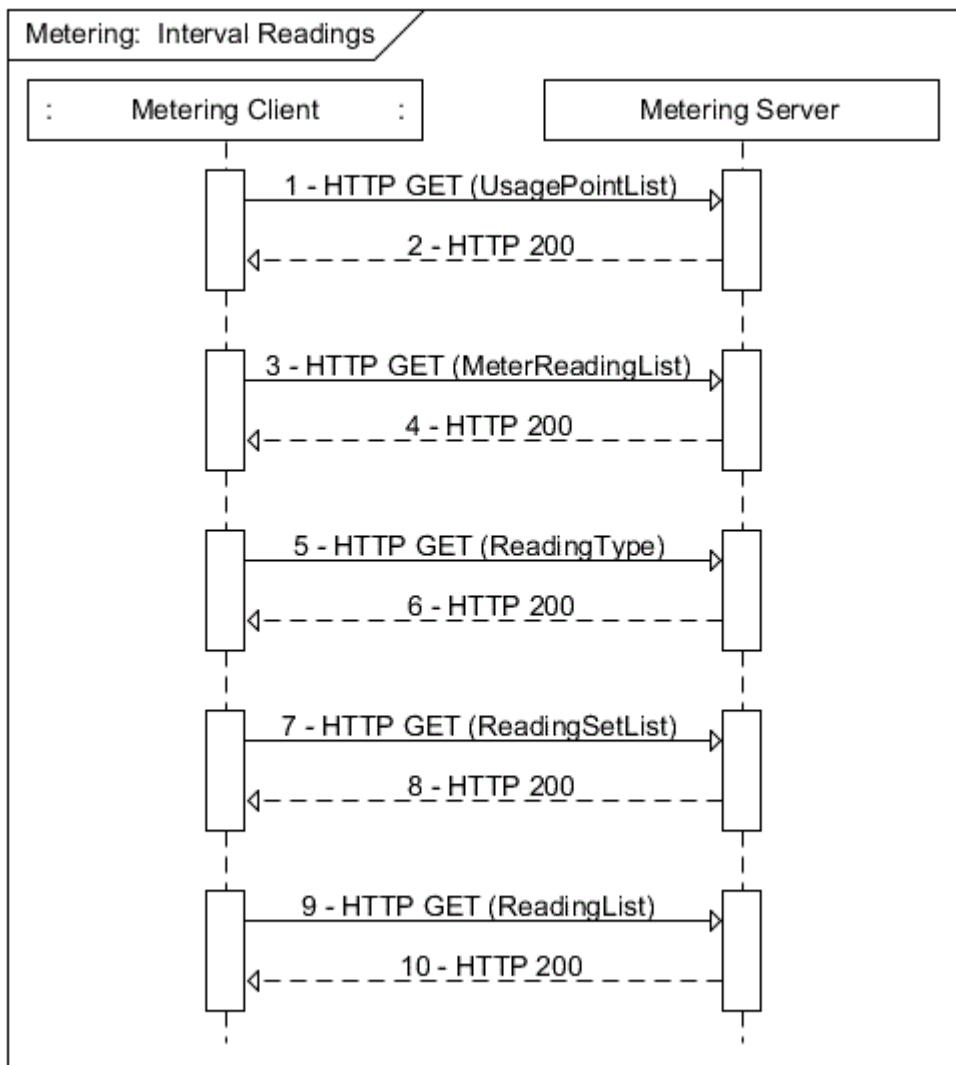
Step	Description
	<pre> <ReadingTypeLink href="/upt/0/mr/3/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/4"> <mRID>1400006CC8</mRID> <description>Instantaneous Reading for Wh</description> <ReadingLink href="/upt/0/mr/4/r"/> <ReadingTypeLink href="/upt/0/mr/4/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/5"> <mRID>1500006CC8</mRID> <description>Instantaneous Reading for VAR's</description> <ReadingLink href="/upt/0/mr/5/r"/> <ReadingTypeLink href="/upt/0/mr/5/rt"/> </MeterReading> </MeterReadingList></pre>
5	<p>Client GETs the ReadingType from the Metering Server.</p> <p>Note: Step 5 and step 6 may be repeated for each MeterReading returned in step 4 to identify the MeterReading of interest by iterating through the MeterReadings returned in step 4.</p> <p>Client sends the following request:</p> <pre>GET /upt/0/mr/0/rt HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
6	<p>Metering Server replies with ReadingType.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingType href="/upt/0/mr/0/rt" xmlns="http://zigbee.org/sep"> <accumulationBehaviour>9</accumulationBehaviour> <commodity>1</commodity> <dataQualifier>0</dataQualifier> <flowDirection>1</flowDirection> <kind>12</kind> <numberOfConsumptionBlocks>1</numberOfConsumptionBlocks> <numberOfTouTiers>4</numberOfTouTiers> <phase>40</phase> <powerOfTenMultiplier>3</powerOfTenMultiplier> <uom>72</uom> </ReadingType></pre> <p>Note: Once the desired ReadingType is identified we proceed to the next step.</p>
7	<p>Client GETs the ReadingSetList from the Metering Server.</p> <p>Note: Because the ReadingSet resources are ordered by their timePeriod earliest time first, we can read the first ReadingSet to get the current values. If a particular historic value is desired you would traverse the ReadingSetList looking for the ReadingSet with the appropriate time stamp.</p> <p>Client sends the following request:</p> <pre>GET /upt/0/mr/0/rs?s=0&l=4 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
8	<p>Metering Server replies with ReadingSetList with the ReadingSet of interest.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK</pre>

Step	Description
	<pre> Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingSetList all="24" href="/upt/0/mr/2/rs" results="4" subscribable="0" xmlns="http://zigbee.org/sep"> <ReadingSet href="/upt/0/mr/2/rs/2"> <mRID>2000006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338842400</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/2/r"/> </ReadingSet> <ReadingSet href="/upt/0/mr/2/rs/4"> <mRID>2200006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338846000</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/4/r"/> </ReadingSet> <ReadingSet href="/upt/0/mr/2/rs/6"> <mRID>2400006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338849600</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/6/r"/> </ReadingSet> <ReadingSet href="/upt/0/mr/2/rs/8"> <mRID>2600006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338853200</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/8/r"/> </ReadingSet> </ReadingSetList> </pre>
9	<p>Client GETs the ReadingList from the Metering Server.</p> <p>Note: Because the Reading resources are ordered by their touTier and remembering the first element is the total Reading, we can read the fourth (index 3) reading to get the current tier three value.</p> <p>Client sends the following request:</p> <pre> GET /upt/0/mr/0/rs/0/r?s=3&l=12 HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>
10	<p>Metering Server replies with ReadingList with the Reading of interest.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingList all="12" href="/upt/0/mr/2/rs/4/r" results="12" xmlns="http://zigbee.org/sep"> <Reading href="/upt/0/mr/2/rs/4/r/0"> <value>1163</value> </pre>

Step	Description
	<pre> <localID>00</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/2"> <value>1162</value> <localID>01</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/4"> <value>1163</value> <localID>02</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/6"> <value>1163</value> <localID>03</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/8"> <value>1163</value> <localID>04</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/a"> <value>1163</value> <localID>05</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/c"> <value>1162</value> <localID>06</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/e"> <value>1163</value> <localID>07</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/10"> <value>1163</value> <localID>08</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/12"> <value>1163</value> <localID>09</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/14"> <value>1162</value> <localID>0A</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/16"> <value>1163</value> <localID>0B</localID> </Reading> </ReadingList> </pre>

7803 16.11 Metering: Interval

7804 This is a use case where a metering function set client (e.g., In-Premises Display) queries for a specific set
 7805 of interval readings from a usage point. For this example, we will assume that the meter is configured for
 7806 5 minute intervals.



7807

7808

Figure 16-11: Metering Interval

7809

Note: In most cases, registration is required to obtain access to metering.

7810

Table 16-13 POX Example: Metering Interval.

Step	Description
1	<p>Client GETs the UsagePointList from the Metering Server.</p> <p>Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.</p> <p>Client sends the following request:</p> <pre>GET /upt HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
2	<p>Metering Server replies with UsagePointList.</p> <p>A typical response looks like:</p> <pre data-bbox="306 386 1139 747"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <UsagePointList all="1" href="/upt" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <UsagePoint href="/upt/0"> <mRID>0B00006CC8</mRID> <description>Usage Point</description> <roleFlags>12</roleFlags> <serviceCategoryKind>0</serviceCategoryKind> <status>1</status> <MeterReadingListLink all="6" href="/upt/0/mr"/> </UsagePoint> </UsagePointList></pre>
3	<p>Client GETs the MeterReadingList from the Metering Server.</p> <p>Note: This and the next 3 steps may be repeated for each page required to read the entire list. For this example, we are requesting up to 10 MeterReadings at a time. Subsequent GETs would increment the "s" query parameter by 10 or however many list items are returned.</p> <p>Client sends the following request:</p> <pre data-bbox="306 967 714 1036"> GET /upt/0/mr?s=0&l=10 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
4	<p>Metering Server replies with up to 10 MeterReadingList instances. Only 6 are returned in this case as indicated by the MeterReadingListLink "all" attribute in step 2.</p> <p>A typical response looks like:</p> <pre data-bbox="306 1174 1225 1826"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <MeterReadingList all="6" href="/upt/0/mr" results="6" subscribable="0" xmlns="http://zigbee.org/sep"> <MeterReading href="/upt/0/mr/0"> <mRID>0C00006CC8</mRID> <description>Cumulative Reading for Wh</description> <ReadingSetListLink all="1" href="/upt/0/mr/0/rs"/> <ReadingTypeLink href="/upt/0/mr/0/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/1"> <mRID>0E00006CC8</mRID> <description>Cumulative Reading for VAR's</description> <ReadingSetListLink all="1" href="/upt/0/mr/1/rs"/> <ReadingTypeLink href="/upt/0/mr/1/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/2"> <mRID>1000006CC8</mRID> <description>Interval Reading for Wh</description> <ReadingSetListLink all="24" href="/upt/0/mr/2/rs"/> <ReadingTypeLink href="/upt/0/mr/2/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/3"> <mRID>1200006CC8</mRID> <description>Interval Reading for VAR's</description> <ReadingSetListLink all="24" href="/upt/0/mr/3/rs"/> </MeterReading></pre>

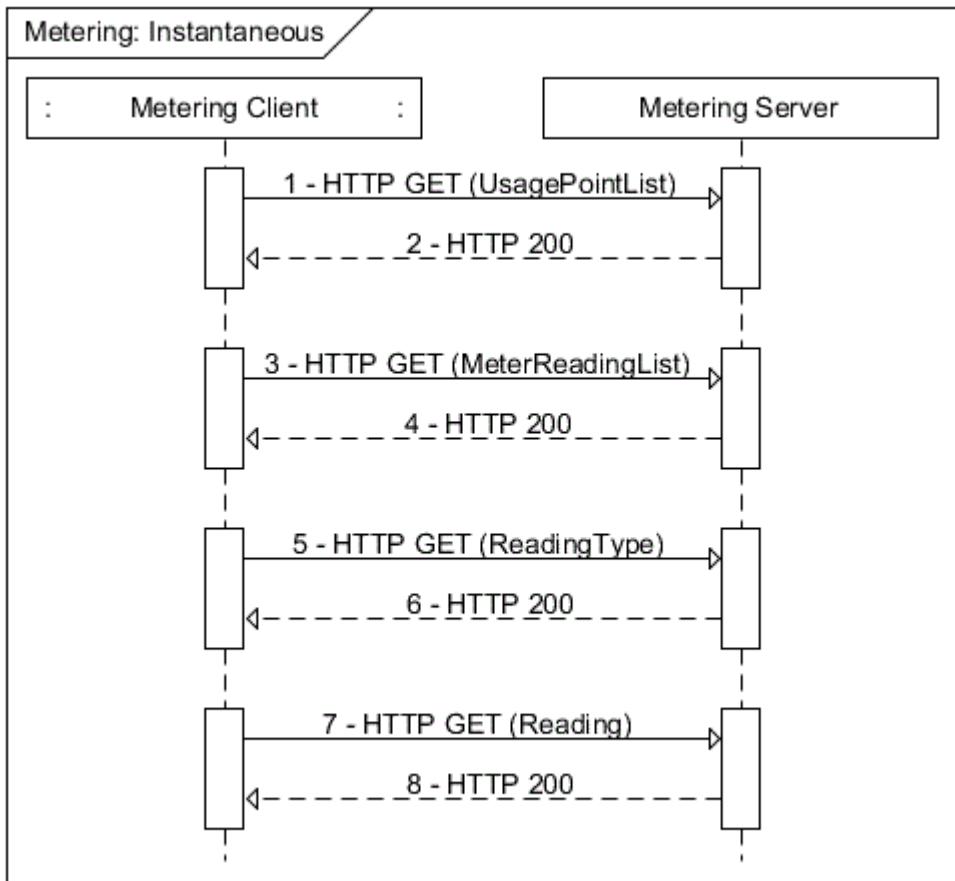
Step	Description
	<pre> <ReadingTypeLink href="/upt/0/mr/3/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/4"> <mRID>1400006CC8</mRID> <description>Instantaneous Reading for Wh</description> <ReadingLink href="/upt/0/mr/4/r"/> <ReadingTypeLink href="/upt/0/mr/4/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/5"> <mRID>00000000000000000000000000000001500006CC8</mRID> <description>Instantaneous Reading for VAR's</description> <ReadingLink href="/upt/0/mr/5/r"/> <ReadingTypeLink href="/upt/0/mr/5/rt"/> </MeterReading> </MeterReadingList> </pre>
5	<p>Client GETs the ReadingType from the Metering Server.</p> <p>Note: Step 5 and step 6 may be repeated for each MeterReading returned in step 4 to identify the MeterReading of interest by iterating through the MeterReadings returned in step 4.</p> <p>Client sends the following request:</p> <pre> GET /upt/0/mr/0/rt?s=0&l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>
6	<p>Metering Server replies with ReadingType.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingType href="/upt/0/mr/0/rt" xmlns="http://zigbee.org/sep"> <accumulationBehaviour>9</accumulationBehaviour> <commodity>1</commodity> <dataQualifier>0</dataQualifier> <flowDirection>1</flowDirection> <kind>12</kind> <numberOfConsumptionBlocks>1</numberOfConsumptionBlocks> <numberOfTouTiers>4</numberOfTouTiers> <phase>40</phase> <powerOfTenMultiplier>3</powerOfTenMultiplier> <uom>72</uom> </ReadingType> </pre> <p>Note: Once the desired ReadingType is identified we proceed to the next step.</p>
7	<p>Client GETs the ReadingSetList from the Metering Server.</p> <p>Note: Because a particular historic sequence of values is desired you would traverse the ReadingSetList looking for the ReadingSet with the timePeriod that encompasses the starting time of the range of intervals of interest.</p> <p>Client sends the following request:</p> <pre> GET /upt/0/mr/2/rs?s=0&l=4 HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>

Step	Description
8	<p>Metering Server replies with ReadingSetList with the ReadingSet of interest.</p> <p>A typical response looks like:</p> <pre data-bbox="306 397 1274 1431"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingSetList all="24" href="/upt/0/mr/2/rs" results="4" subscribable="0" xmlns="http://zigbee.org/sep"> <ReadingSet href="/upt/0/mr/2/rs/2"> <mRID>2000006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338842400</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/2/r"/> </ReadingSet> <ReadingSet href="/upt/0/mr/2/rs/4"> <mRID>2200006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338846000</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/4/r"/> </ReadingSet> <ReadingSet href="/upt/0/mr/2/rs/6"> <mRID>2400006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338849600</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/6/r"/> </ReadingSet> <ReadingSet href="/upt/0/mr/2/rs/8"> <mRID>2600006CC8</mRID> <description>Reading Set for WHrs</description> <timePeriod> <duration>3600</duration> <start>1338853200</start> </timePeriod> <ReadingListLink all="12" href="/upt/0/mr/2/rs/8/r"/> </ReadingSet> </ReadingSetList></pre>
9	<p>Once the ReadingSet that encompasses the starting time is identified, the client would GET the ReadingList from the Metering Server.</p> <p>Note: To identify the interval that has the desired start time the client would GET the reading set.</p> <p>Client sends the following request:</p> <pre data-bbox="306 1611 829 1685"> GET /upt/0/mr/2/rs/4/r?s=0&l=12 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
10	<p>Metering Server replies with ReadingList.</p> <p>A typical response looks like:</p> <pre data-bbox="306 1780 739 1856"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength}</pre>

Step	Description
	<pre> <ReadingList all="12" href="/upt/0/mr/2/rs/4/r" results="12" xmlns="http://zigbee.org/sep"> <Reading href="/upt/0/mr/2/rs/4/r/0"> <value>1163</value> <localID>00</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/2"> <value>1162</value> <localID>01</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/4"> <value>1163</value> <localID>02</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/6"> <value>1163</value> <localID>03</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/8"> <value>1163</value> <localID>04</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/a"> <value>1163</value> <localID>05</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/c"> <value>1162</value> <localID>06</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/e"> <value>1163</value> <localID>07</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/10"> <value>1163</value> <localID>08</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/12"> <value>1163</value> <localID>09</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/14"> <value>1162</value> <localID>0A</localID> </Reading> <Reading href="/upt/0/mr/2/rs/4/r/16"> <value>1163</value> <localID>0B</localID> </Reading> </ReadingList> </pre> <p>The client would then walk the list starting at the "start" time in the timePeriod of the ReadingSet and adding, for readings that specify their timePeriod, the duration or for Readings that don't specify their timePeriod the intervalLength from ReadingType. If all intervals of interest are not contained in the current reading set then we repeat the last two steps to get the additional data. If the information gathered in step 7 is exhausted then you need to loop back to step 7 to GET the next set of ReadingSets.</p>

7811 **16.12 Metering: Instantaneous**

7812 This is a use case where a metering function set client (e.g., In-Premises Display) queries an
7813 instantaneous Watts reading from a usage point.



7814

7815

Figure 16-12: Metering Instantaneous

7816 Note: In most cases, registration is required to obtain access to metering.

7817 **Table 16-14 POX Example: Metering Instantaneous.**

Step	Description
1	<p>Client GETs the UsagePointList from the Metering Server</p> <p>Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.</p> <p>Client sends the following request:</p> <pre>GET /upt HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

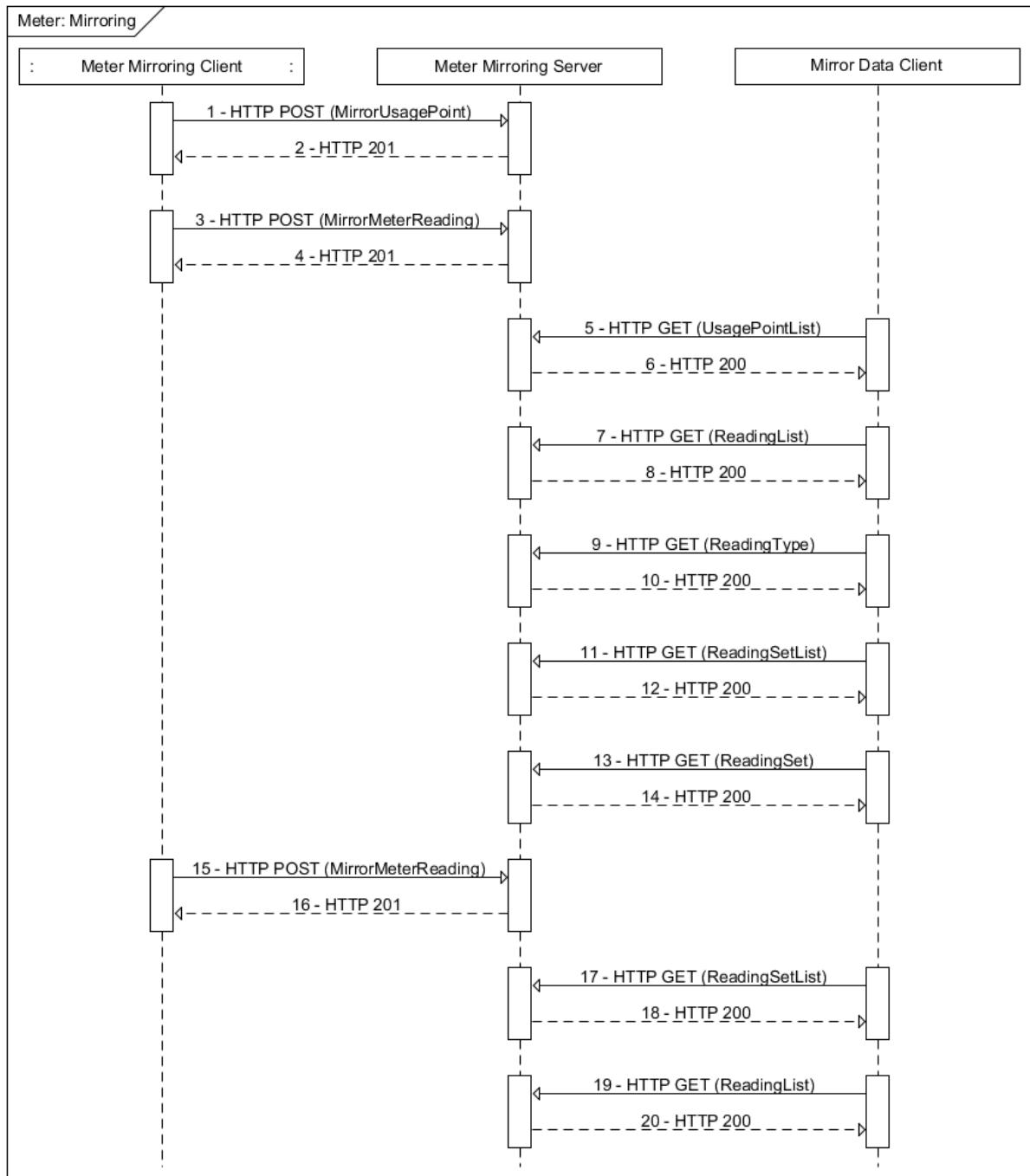
Step	Description
2	<p>Metering Server replies with UsagePointList.</p> <p>A typical response looks like:</p> <pre data-bbox="306 386 1139 747"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <UsagePointList all="1" href="/upt" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <UsagePoint href="/upt/0"> <mRID>0B00006CC8</mRID> <description>Usage Point</description> <roleFlags>12</roleFlags> <serviceCategoryKind>0</serviceCategoryKind> <status>1</status> <MeterReadingListLink all="6" href="/upt/0/mr"/> </UsagePoint> </UsagePointList></pre>
3	<p>Client GETs the MeterReadingList from the Metering Server.</p> <p>Note: This and the next 3 steps may be repeated for each page required to read the entire list. For this example, we are requesting up to 10 MeterReadings at a time. Subsequent GETs would increment the "s" query parameter by 10 or however many list items are returned.</p> <p>Client sends the following request:</p> <pre data-bbox="306 967 714 1036"> GET /upt/0/mr?s=0&l=10 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
4	<p>Metering Server replies with up to 10 MeterReading instances. Only 6 are returned in this case as indicated by the MeterReadingListLink "all" attribute in step 2.</p> <p>A typical response looks like:</p> <pre data-bbox="306 1174 1225 1826"> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <MeterReadingList all="6" href="/upt/0/mr" results="6" subscribable="0" xmlns="http://zigbee.org/sep"> <MeterReading href="/upt/0/mr/0"> <mRID>0C00006CC8</mRID> <description>Cumulative Reading for Wh</description> <ReadingSetListLink all="1" href="/upt/0/mr/0/rs"/> <ReadingTypeLink href="/upt/0/mr/0/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/1"> <mRID>0E00006CC8</mRID> <description>Cumulative Reading for VAR's</description> <ReadingSetListLink all="1" href="/upt/0/mr/1/rs"/> <ReadingTypeLink href="/upt/0/mr/1/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/2"> <mRID>1000006CC8</mRID> <description>Interval Reading for Wh</description> <ReadingSetListLink all="24" href="/upt/0/mr/2/rs"/> <ReadingTypeLink href="/upt/0/mr/2/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/3"> <mRID>1200006CC8</mRID> <description>Interval Reading for VAR's</description> <ReadingSetListLink all="24" href="/upt/0/mr/3/rs"/> </MeterReading></pre>

Step	Description
	<pre> <ReadingTypeLink href="/upt/0/mr/3/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/4"> <mRID>1400006CC8</mRID> <description>Instantaneous Reading for Wh</description> <ReadingLink href="/upt/0/mr/4/r"/> <ReadingTypeLink href="/upt/0/mr/4/rt"/> </MeterReading> <MeterReading href="/upt/0/mr/5"> <mRID>00000000000000000000000000000001500006CC8</mRID> <description>Instantaneous Reading for VAR's</description> <ReadingLink href="/upt/0/mr/5/r"/> <ReadingTypeLink href="/upt/0/mr/5/rt"/> </MeterReading> </MeterReadingList> </pre>
5	<p>Client GETs the ReadingType from the Metering Server.</p> <p>Note: Step 5 and step 6 may be repeated for each MeterReading returned in step 4 to identify the MeterReading of interest by iterating through the MeterReadings returned in step 4.</p> <p>Client sends the following request:</p> <pre> GET /upt/0/mr/4/rt HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>
6	<p>Metering Server replies with ReadingType.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingType href="/upt/0/mr/4/rt" xmlns="http://zigbee.org/sep"> <accumulationBehaviour>12</accumulationBehaviour> <commodity>1</commodity> <dataQualifier>0</dataQualifier> <flowDirection>1</flowDirection> <kind>12</kind> <numberOfConsumptionBlocks>0</numberOfConsumptionBlocks> <numberOfTouTiers>0</numberOfTouTiers> <phase>40</phase> <powerOfTenMultiplier>3</powerOfTenMultiplier> <uom>38</uom> </ReadingType> </pre> <p>Note: Once the desired ReadingType is identified we proceed to the next step.</p>
7	<p>Client GETs the Reading from the Metering Server.</p> <p>Note: Because the instantaneous value is in the resource indicated in the ReadingLink of MeterReading we can read that resource.</p> <p>Client sends the following request:</p> <pre> GET /upt/0/mr/4/r HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>

Step	Description
8	<p>Metering Server replies with the Reading.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <Reading href="/upt/0/mr/4/r" xmlns="http://zigbee.org/sep"> <value>14</value> </Reading></pre> <p>Note: Subsequent reads of this URI will return more recent data. If a GET fails on this resource then this entire procedure would be repeated to reestablish the correct URI.</p>

7818 **16.13 Metering: Mirroring**

7819 This is a use case where a mirror metering function set client (e.g., a gas meter) POSTs first its general
 7820 information and then its data. It also shows a client of the gas meter data retrieving the most recent 24
 7821 intervals of data. An assumption is made that prior to this sequence the mirror client has discovered the
 7822 URI of the appropriate Meter Mirroring Server.



7823

7824

Figure 16-13: Meter Mirroring

7825 Note: In most cases, registration is required to obtain access to metering and meter mirroring.

7826 **Table 16-15 POX Example: Meter Mirroring.**

Step	Description
------	-------------

Step	Description
1	<p>Meter Mirroring Client POSTs a MirrorUsagePoint to the Mirror Metering Server. It is including the current consumption value. This could have been done in a separate POST to the resultant MirrorUsagePoint.</p> <p>Note: It passes two ReadingType definitions, one for Summation and one for Interval data.</p> <p>Client sends the following request:</p> <pre> POST /mup HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <MirrorUsagePoint xmlns="http://zigbee.org/sep"> <mRID>0600006CC8</mRID> <description>Gas Mirroring</description> <roleFlags>13</roleFlags> <serviceCategoryKind>1</serviceCategoryKind> <status>1</status> <deviceLFDI>3E4F45AB31EDFE5B67E343E5E4562E31984E23E5</deviceLFDI> <MirrorMeterReading> <mRID>0700006CC8</mRID> <description>Cumulative Reading for Gas</description> <Reading> <value>125</value> </Reading> <ReadingType> <accumulationBehaviour>9</accumulationBehaviour> <commodity>7</commodity> <dataQualifier>0</dataQualifier> <flowDirection>1</flowDirection> <powerOfTenMultiplier>3</powerOfTenMultiplier> < uom>119</ uom> </ReadingType> </MirrorMeterReading> <MirrorMeterReading> <mRID>0800006CC8</mRID> <description>Interval Readings for Gas</description> <ReadingType> <accumulationBehaviour>4</accumulationBehaviour> <commodity>7</commodity> <dataQualifier>0</dataQualifier> <flowDirection>1</flowDirection> <powerOfTenMultiplier>3</powerOfTenMultiplier> < uom>119</ uom> </ReadingType> </MirrorMeterReading> </MirrorUsagePoint></pre>
2	<p>Mirror Metering Server creates a MirrorUsagePoint and UsagePoint with the data supplied in the MirrorUsagePoint, adds it to its MirrorUsagePointList and then replies with the URI of the MirrorUsagePoint.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 201 Created Content-Length: 0 Location: /mup/0</pre>

Step	Description
3	<p>Meter Mirroring Client POSTs a MirrorMeterReading to the Mirror Metering Server.</p> <p>Client sends the following request:</p> <pre> POST /mup/0 HTTP/1.1 Host: {hostname} Content-Type: application/sep+xml Content-Length: {contentLength} <MirrorMeterReading xmlns="http://zigbee.org/sep"> <mRID>0800006CC8</mRID> <MirrorReadingSet> <mRID>0900006CC8</mRID> <timePeriod> <duration>86400</duration> <start>1341579365</start> </timePeriod> <Reading> <value>9</value> <localID>00</localID> </Reading> <Reading> <value>11</value> <localID>01</localID> </Reading> <Reading> <value>10</value> <localID>02</localID> </Reading> <Reading> <value>13</value> <localID>03</localID> </Reading> <Reading> <value>12</value> <localID>04</localID> </Reading> <Reading> <value>11</value> <localID>05</localID> </Reading> <Reading> <value>10</value> <localID>06</localID> </Reading> <Reading> <value>16</value> <localID>07</localID> </Reading> <Reading> <value>9</value> <localID>08</localID> </Reading> <Reading> <value>7</value> <localID>09</localID> </Reading> <Reading> <value>6</value> <localID>0A</localID> </Reading> <Reading> <value>5</value> <localID>0B</localID> </Reading> </MirrorReadingSet> </MirrorMeterReading></pre>

Step	Description
	<pre> <Reading> <value>8</value> <localID>0C</localID> </Reading> <Reading> <value>9</value> <localID>0D</localID> </Reading> <Reading> <value>10</value> <localID>0E</localID> </Reading> <Reading> <value>12</value> <localID>0F</localID> </Reading> <Reading> <value>14</value> <localID>10</localID> </Reading> <Reading> <value>13</value> <localID>11</localID> </Reading> <Reading> <value>11</value> <localID>12</localID> </Reading> <Reading> <value>7</value> <localID>13</localID> </Reading> <Reading> <value>8</value> <localID>14</localID> </Reading> <Reading> <value>10</value> <localID>15</localID> </Reading> <Reading> <value>10</value> <localID>16</localID> </Reading> <Reading> <value>10</value> <localID>17</localID> </Reading> </MirrorReadingSet> </MirrorMeterReading> </pre>
4	<p>Mirror Metering Server creates a MirrorMeterReading with the data supplied in the MirrorUsagePoint and then replies with the URI of the MirrorMeterReading.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 201 Created Content-Length: 0 Location: /upt/1/mr </pre> <p>Note: Steps 1-4 could be combined into 2 steps by including the initial interval reading set data in the initial MirrorUsagePoint POST.</p>
5	<p>Mirror Data Client GETs the UsagePointList from the Metering Server</p>

Step	Description
	<p>Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.</p> <p>Client sends the following request:</p> <pre>GET /upt HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
6	<p>Mirror Metering Server replies with UsagePointList.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <UsagePointList all="2" href="/upt" results="2" subscribable="0" xmlns="http://zigbee.org/sep"> <UsagePoint href="/upt/0"> <mRID>0B00006CC8</mRID> <description>Usage Point</description> <roleFlags>12</roleFlags> <serviceCategoryKind>0</serviceCategoryKind> <status>1</status> <MeterReadingListLink all="6" href="/upt/0/mr"/> </UsagePoint> <UsagePoint href="/upt/1"> <mRID>0C00006CC8</mRID> <description>Usage Point</description> <roleFlags>13</roleFlags> <serviceCategoryKind>1</serviceCategoryKind> <status>1</status> <MeterReadingListLink all="2" href="/upt/1/mr"/> </UsagePoint> </UsagePointList></pre>
7	<p>Mirror Data Client GETs the MeterReadingList from the Mirror Metering Server.</p> <p>Note: We will choose /upt/1 because its role flags indicate it is a mirror.</p> <p>Note: This and the next 3 steps may be repeated for each page required to read the entire list. For this example, we are requesting up to 10 MeterReadings at a time. Subsequent GETs would increment the "s" query parameter by 10 or however many list items are returned.</p> <p>Client sends the following request:</p> <pre>GET /upt/1/mr?s=0&l=10 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
8	<p>Mirror Metering Server replies with up to 10 MeterReadingList instances. Only 2 are returned in this case as indicated by the MeterReadingListLink "all" attribute in step 6.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <MeterReadingList all="2" href="/upt/1/mr" results="2" subscribable="0" xmlns="http://zigbee.org/sep"> <MeterReading href="/upt/1/mr/0"> <mRID>0700006CC8</mRID> <description>Cumulative Reading for Gas</description> <ReadingLink href="/upt/1/mr/0/r"/></pre>

Step	Description
	<pre> <ReadingTypeLink href="/upt/1/mr/0/rt"/> </MeterReading> <MeterReading href="/upt/1/mr/1"> <mRID>0800006CC8</mRID> <description> Interval Readings for Gas</description> <ReadingSetListLink all="1" href="/upt/1/mr/1/rs"/> <ReadingTypeLink href="/upt/1/mr/1/rt"/> </MeterReading> </MeterReadingList></pre>
9	<p>Mirror Data Client GETs the ReadingType from the Mirror Metering Server.</p> <p>Note: Step 9 and step 10 may be repeated for each MeterReading returned in step 4 to identify the MeterReading of interest by iterating through the MeterReadings returned in step 4.</p> <p>Client sends the following request:</p> <pre> GET /upt/1/mr/1/rt HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
10	<p>Mirror Metering Server replies with ReadingType.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingType href="/upt/1/mr/1/rt" xmlns="http://zigbee.org/sep"> <accumulationBehaviour>4</accumulationBehaviour> <commodity>7</commodity> <flowDirection>1</flowDirection> <powerOfTenMultiplier>3</powerOfTenMultiplier> <uom>119</uom> </ReadingType></pre> <p>Note: Once the desired ReadingType is identified we proceed to the next step.</p>
11	<p>Mirror Data Client GETs the ReadingSetList from the Metering Server.</p> <p>Note: Because the ReadingSet resources are ordered by their timePeriod earliest time first, we can read the first ReadingSet to get the current values. If a particular historic value is desired you would traverse the ReadingSetList looking for the ReadingSet with the appropriate time stamp.</p> <p>Client sends the following request:</p> <pre> GET /upt/1/mr/1/rs?s=0&l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
12	<p>Mirror Metering Server replies with ReadingSetList with the ReadingSet of interest.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingSetList all="1" href="/upt/1/mr/1/rs" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <ReadingSet href="/upt/1/mr/1/rs/32"> <mRID>2000006CC8</mRID> <timePeriod> <duration>86400</duration> <start>1341579365</start></pre>

Step	Description
	<pre> </timePeriod> <ReadingListLink all="24" href="/upt/1/mr/1/rs/32/r"/> </ReadingSet> </ReadingSetList> </pre>
13	<p>Mirror Data Client GETs the ReadingList from the Mirror Metering Server.</p> <p>Client sends the following request:</p> <pre> GET /upt/1/mr/1/rs/32/r?s=0&l=24 HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>
14	<p>Mirror Metering Server replies with ReadingList with the Reading of interest.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingList all="24" href="/upt/1/mr/1/rs/32/r" results="24" xmlns="http://zigbee.org/sep"> <Reading href="/upt/1/mr/1/rs/32/r/4"> <value>9</value> <localID>00</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/5"> <value>11</value> <localID>01</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/6"> <value>10</value> <localID>02</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/7"> <value>13</value> <localID>03</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/8"> <value>12</value> <localID>04</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/9"> <value>11</value> <localID>05</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/a"> <value>10</value> <localID>06</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/b"> <value>16</value> <localID>07</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/c"> <value>9</value> <localID>08</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/d"> <value>7</value> <localID>09</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/e"> <value>6</value> </pre>

Step	Description
	<pre> <localID>0A</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/f"> <value>5</value> <localID>0B</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/10"> <value>8</value> <localID>0C</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/11"> <value>9</value> <localID>0D</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/12"> <value>10</value> <localID>0E</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/13"> <value>12</value> <localID>0F</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/14"> <value>14</value> <localID>10</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/15"> <value>13</value> <localID>11</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/16"> <value>11</value> <localID>12</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/17"> <value>7</value> <localID>13</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/0"> <value>8</value> <localID>14</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/1"> <value>10</value> <localID>15</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/2"> <value>10</value> <localID>16</localID> </Reading> <Reading href="/upt/1/mr/1/rs/32/r/3"> <value>10</value> <localID>17</localID> </Reading> </ReadingList> </pre>
15	<p>The next day, the Meter Mirroring Client POSTs a MirrorMeterReadingList with MirrorMeterReadings to the Mirror Metering Server. The first MirrorMeterReading is the consumption (cumulative) value and the second MirrorMeterReading is a set of interval data.</p> <p>Client sends the following request:</p> <pre>POST /mup/0 HTTP/1.1 Host: {hostname}</pre>

Step	Description
	<pre> Content-Type: application/sep+xml Content-Length: {contentLength} <MirrorMeterReadingList xmlns="http://zigbee.org/sep"> <MirrorMeterReading> <mRID>0700006CC8</mRID> <Reading> <value>574</value> </Reading> </MirrorMeterReading> <MirrorMeterReading> <mRID>0800006CC8</mRID> <MirrorReadingSet> <mRID>0900006CC8</mRID> <timePeriod> <duration>86400</duration> <start>1341665765</start> </timePeriod> <Reading> <value>9</value> <localID>00</localID> </Reading> <Reading> <value>12</value> <localID>01</localID> </Reading> <Reading> <value>10</value> <localID>02</localID> </Reading> <Reading> <value>13</value> <localID>03</localID> </Reading> <Reading> <value>11</value> <localID>04</localID> </Reading> <Reading> <value>11</value> <localID>05</localID> </Reading> <Reading> <value>10</value> <localID>06</localID> </Reading> <Reading> <value>12</value> <localID>07</localID> </Reading> <Reading> <value>9</value> <localID>08</localID> </Reading> <Reading> <value>7</value> <localID>09</localID> </Reading> <Reading> <value>6</value> <localID>0A</localID> </Reading> <Reading> <value>5</value> <localID>0B</localID> </Reading> </MirrorReadingSet> </MirrorMeterReading> </MirrorMeterReadingList> </pre>

Step	Description
	<pre> </Reading> <Reading> <value>8</value> <localID>0C</localID> </Reading> <Reading> <value>9</value> <localID>0D</localID> </Reading> <Reading> <value>10</value> <localID>0E</localID> </Reading> <Reading> <value>12</value> <localID>0F</localID> </Reading> <Reading> <value>14</value> <localID>10</localID> </Reading> <Reading> <value>13</value> <localID>11</localID> </Reading> <Reading> <value>11</value> <localID>12</localID> </Reading> <Reading> <value>7</value> <localID>13</localID> </Reading> <Reading> <value>8</value> <localID>14</localID> </Reading> <Reading> <value>10</value> <localID>15</localID> </Reading> <Reading> <value>10</value> <localID>16</localID> </Reading> <Reading> <value>10</value> <localID>17</localID> </Reading> </MirrorReadingSet> </MirrorMeterReading> </pre>
16	<p>Mirror Metering Server copies the data supplied into the corresponding UsagePoint and then replies with the URI of the MeterReadingList of that UsagePoint.</p> <p>A typical response looks like:</p> <pre> HTTP/1.1 201 Created Content-Length: 0 Location: /upt/1/mr </pre>

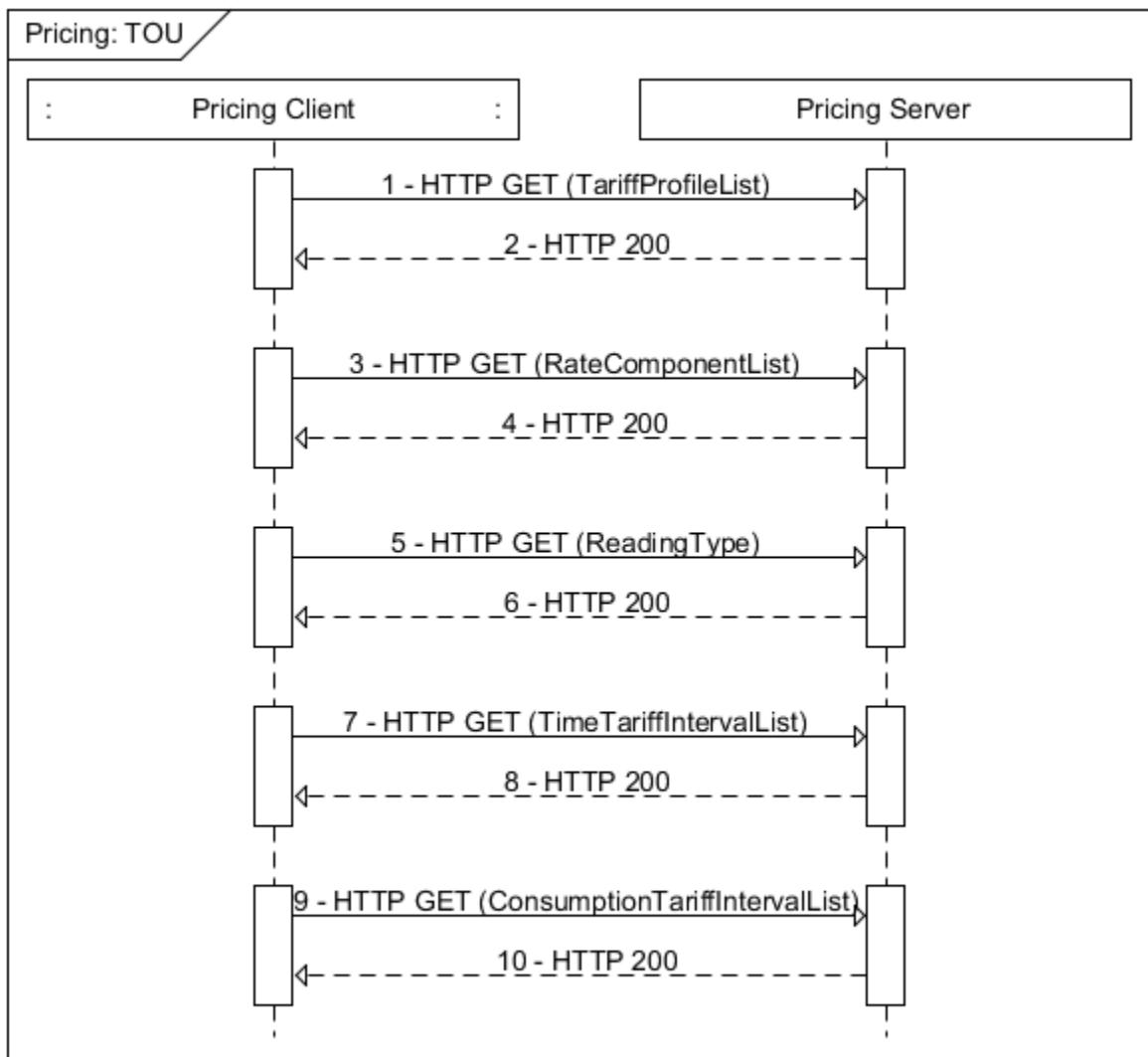
Step	Description
17	<p>Mirror Data Client GETs the ReadingSetList from the Mirror Metering Server.</p> <p>Note: Because the Client cached the URI of the reading set list, it can skip ahead to this step.</p> <p>Client sends the following request:</p> <pre>GET /upt/1/mr/1/rs?s=0&l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
18	<p>Mirror Metering Server replies with ReadingSetList with the ReadingSet of interest.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingSetList all="1" href="/upt/1/mr/1/rs" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <ReadingSet href="/upt/1/mr/1/rs/33"> <mRID>A000006CC8</mRID> <timePeriod> <duration>86400</duration> <start>1341665765</start> </timePeriod> <ReadingListLink all="24" href="/upt/1/mr/1/rs/33/r"/> </ReadingSet> </ReadingSetList></pre>
19	<p>Mirror Data Client GETs the ReadingList from the Mirror Metering Server.</p> <p>Client sends the following request:</p> <pre>GET /upt/1/mr/1/rs/33/r?s=0&l=24 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
20	<p>Mirror Metering Server replies with ReadingList with the Reading of interest.</p> <p>A typical response looks like:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingList all="12" href="/upt/1/mr/1/rs/33/r" results="24" xmlns="http://zigbee.org/sep"> <Reading href="/upt/1/mr/1/rs/33/r/4"> <value>9</value> <localID>00</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/5"> <value>12</value> <localID>01</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/6"> <value>10</value> <localID>02</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/7"> <value>13</value> <localID>03</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/8"> <value>11</value> <localID>04</localID></pre>

Step	Description
	<pre> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/9"> <value>11</value> <localID>05</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/a"> <value>10</value> <localID>06</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/b"> <value>12</value> <localID>07</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/c"> <value>9</value> <localID>08</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/d"> <value>7</value> <localID>09</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/e"> <value>6</value> <localID>0A</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/f"> <value>5</value> <localID>0B</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/10"> <value>8</value> <localID>0C</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/11"> <value>9</value> <localID>0D</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/12"> <value>10</value> <localID>0E</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/13"> <value>12</value> <localID>0F</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/14"> <value>14</value> <localID>10</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/15"> <value>13</value> <localID>11</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/16"> <value>11</value> <localID>12</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/17"> <value>7</value> <localID>13</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/0"> <value>8</value> <localID>14</localID> </Reading> </pre>

Step	Description
	<Reading href="/upt/1/mr/1/rs/33/r/1"> <value>10</value> <localID>15</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/2"> <value>10</value> <localID>16</localID> </Reading> <Reading href="/upt/1/mr/1/rs/33/r/3"> <value>10</value> <localID>17</localID> </Reading> </ReadingList>

7827 16.14 **Pricing: Time of Use**

7828 This flow diagram describes pricing client device obtaining pricing information.



7829

7830

Figure 16-14: Pricing Time of Use

7831 Note: In most cases, registration is required to obtain access to pricing, and the client is directed to a
 7832 specific *TariffProfile* through a Link in the *FunctionSetAssignments* found in their *EndDevice*
 7833 *FunctionSetAssignmentsListLink*, or from a different function set such as Billing.

7834 **Table 16-16 POX Example: Pricing TOU.**

Step	Description
1	<p>Client GETs the TariffProfile from the Pricing Server.</p> <p>Client sends the following request:</p> <pre>GET /tp/3 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
2	<p>Pricing server responds with the TariffProfile.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <TariffProfile href="/tp/3" xmlns="http://zigbee.org/sep"> <mRID>799794f4620b17e00000e566</mRID> <description>PEV TOU Rate</description> <currency>840</currency> <pricePowerOfTenMultiplier>-6</pricePowerOfTenMultiplier> <primacy>0</primacy> <rateCode>TOU-D-PEV Baseline 6</rateCode> <RateComponentListLink all="1" href="/tp/3/rc"/> <serviceCategoryKind>0</serviceCategoryKind> </TariffProfile></pre>
3	<p>Client GETs the RateComponentList from the Pricing Server.</p> <p>Client sends the following request:</p> <pre>GET /tp/3/rc?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
4	<p>Pricing server responds with the RateComponentList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <RateComponentList all="1" href="/tp/3/rc" results="1" xmlns="http://zigbee.org/sep"> <RateComponent href="/tp/3/rc/3"> <mRID>fc000b07143d24fc0000e566</mRID> <description>TOU-D-PEV</description> <ActiveTimeTariffIntervalListLink all="0" href="/tp/3/rc/3/acttti"/> <flowRateEndLimit> <multiplier>0</multiplier> <unit>38</unit> <value>400</value> </flowRateEndLimit> <flowRateStartLimit> <multiplier>0</multiplier> <unit>38</unit> <value>0</value> </flowRateStartLimit> <ReadingTypeLink href="/rt/1"/> <roleFlags>12</roleFlags> <TimeTariffIntervalListLink all="5" href="/tp/3/rc/3/tti"/> </RateComponent> </RateComponentList></pre>
5	<p>Client GETs the ReadingType from the Pricing Server.</p> <p>Client sends the following request:</p> <pre>GET /rt/1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
6	<p>Pricing server responds with the ReadingType.</p> <p>Server sends the following response:</p>

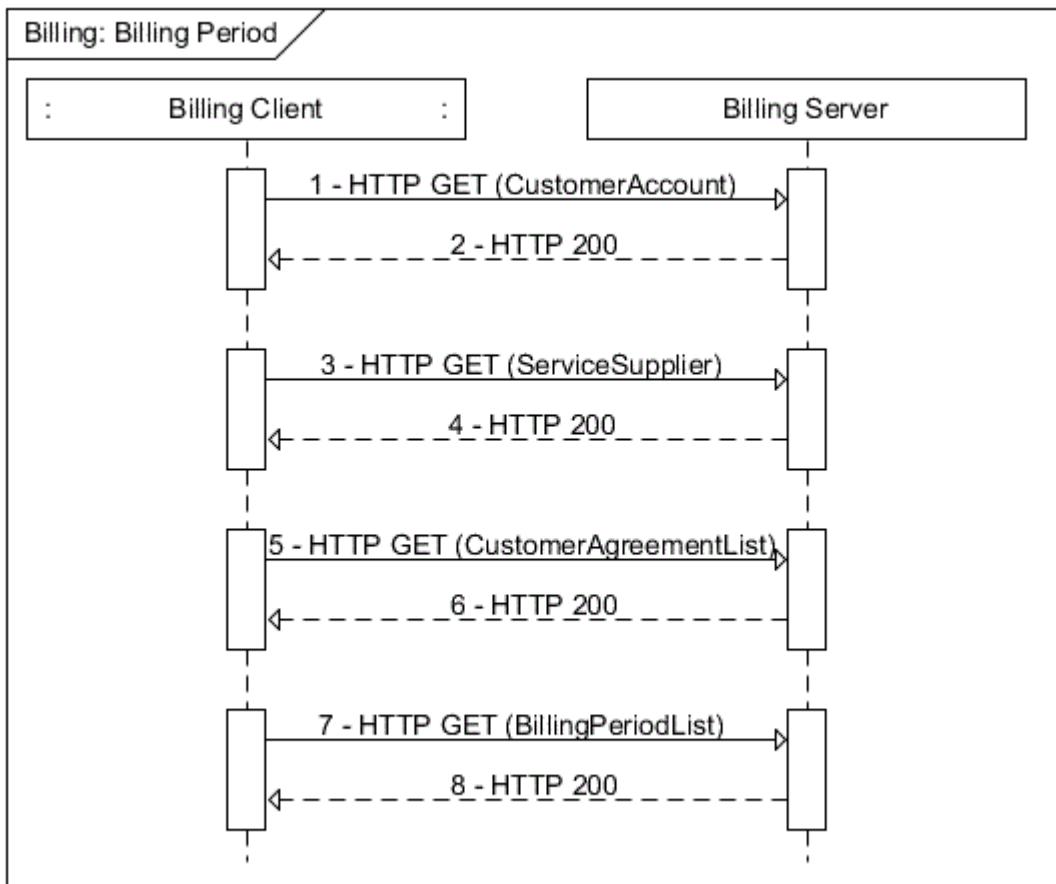
Step	Description
	<p>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength}</p> <pre><ReadingType href="/rt/1" xmlns="http://zigbee.org/sep"> <accumulationBehaviour>4</accumulationBehaviour> <commodity>1</commodity> <dataQualifier>12</dataQualifier> <flowDirection>1</flowDirection> <intervalLength>3600</intervalLength> <kind>12</kind> <numberOfConsumptionBlocks>1</numberOfConsumptionBlocks> <numberOfTouTiers>3</numberOfTouTiers> <phase>0</phase> <powerOfTenMultiplier>3</powerOfTenMultiplier> <tieredConsumptionBlocks>false</tieredConsumptionBlocks> <uom>72</uom> </ReadingType></pre>
7	<p>Client GETs the TimeTariffIntervalList from the Pricing Server.</p> <p>Client sends the following request:</p> <pre>GET /tp/3/rc/3/tti?l=5 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
8	<p>Pricing server responds with the TimeTariffIntervalList.</p> <p>Server sends the following response:</p> <p>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength}</p> <pre><TimeTariffIntervalList all="5" href="/tp/3/rc/3/tti" results="5" subscribable="1" xmlns="http://zigbee.org/sep"> <TimeTariffInterval href="/tp/3/rc/3/tti/5" subscribable="1"> <mRID>f06fa23dc0a0f650000e566</mRID> <description>Off-Peak 1</description> <creationTime>1357430400</creationTime> <EventStatus> <currentStatus>0</currentStatus> <dateTime>1357430400</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <duration>28800</duration> <start>1357516800</start> </interval> <randomizeDuration>300</randomizeDuration> <randomizeStart>300</randomizeStart> <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/5/cti"/> <touTier>1</touTier> </TimeTariffInterval> <TimeTariffInterval href="/tp/3/rc/3/tti/6" subscribable="1"> <mRID>41fc7c07e16820770000e566</mRID> <description>Mid-Peak 1</description> <creationTime>1357430400</creationTime> <EventStatus> <currentStatus>0</currentStatus> <dateTime>1357430400</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval></pre>

Step	Description
	<pre> <duration>14400</duration> <start>1357545600</start> </interval> <randomizeDuration>300</randomizeDuration> <randomizeStart>300</randomizeStart> <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/6/cti"/> <touTier>2</touTier> </TimeTariffInterval> <TimeTariffInterval href="/tp/3/rc/3/tti/7" subscribable="1"> <mRID>63eed7b30c1c87a40000e566</mRID> <description>On-Peak</description> <creationTime>1357430400</creationTime> <EventStatus> <currentStatus>0</currentStatus> <dateTime>1357430400</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <duration>21600</duration> <start>1357552800</start> </interval> <randomizeDuration>300</randomizeDuration> <randomizeStart>300</randomizeStart> <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/7/cti"/> <touTier>3</touTier> </TimeTariffInterval> <TimeTariffInterval href="/tp/3/rc/3/tti/8" subscribable="1"> <mRID>9b04f0713e9212d90000e566</mRID> <description>Mid-Peak 2</description> <creationTime>1357430400</creationTime> <EventStatus> <currentStatus>0</currentStatus> <dateTime>1357430400</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <duration>18000</duration> <start>1357574400</start> </interval> <randomizeDuration>300</randomizeDuration> <randomizeStart>300</randomizeStart> <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/8/cti"/> <touTier>2</touTier> </TimeTariffInterval> <TimeTariffInterval href="/tp/3/rc/3/tti/9" subscribable="1"> <mRID>c13c8755dc39b595000e566</mRID> <description>Off-Peak 2</description> <creationTime>1357430400</creationTime> <EventStatus> <currentStatus>0</currentStatus> <dateTime>1357430400</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <duration>10800</duration> <start>1357592400</start> </interval> <randomizeDuration>300</randomizeDuration> <randomizeStart>300</randomizeStart> <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/9/cti"/> <touTier>1</touTier> </TimeTariffInterval> </TimeTariffIntervalList> </pre>
9	Client GETs a ConsumptionTariffIntervalList from the Pricing Server (note that there is a

Step	Description
	<p>ConsumptionTariffIntervalList for each TimeTariffInterval, but only one is shown below.)</p> <p>Client sends the following request:</p> <pre>GET /tp/3/rc/3/tti/5/cti?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
10	<p>Pricing server responds with the ConsumptionTariffIntervalList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ConsumptionTariffIntervalList all="1" href="/tp/3/rc/3/tti/5/cti" results="1" xmlns="http://zigbee.org/sep"> <ConsumptionTariffInterval href="/tp/3/rc/3/tti/5/cti/1"> <consumptionBlock>1</consumptionBlock> <price>113000</price> <startValue>0</startValue> </ConsumptionTariffInterval> </ConsumptionTariffIntervalList></pre>

7835 **16.15 Billing: Billing Period**

7836 This flow diagram describes a billing client device obtaining billing information including billing period.



7837

7838

Figure 16-15: Billing Period

7839 Note: In most cases, registration is required to obtain access to billing, and the client is directed to a
 7840 specific *CustomerAccount* through a Link in the *FunctionSetAssignments* found in their *EndDevice*
 7841 *FunctionSetAssignmentsListLink*.

7842 **Table 16-17 POX Example: Billing Period.**

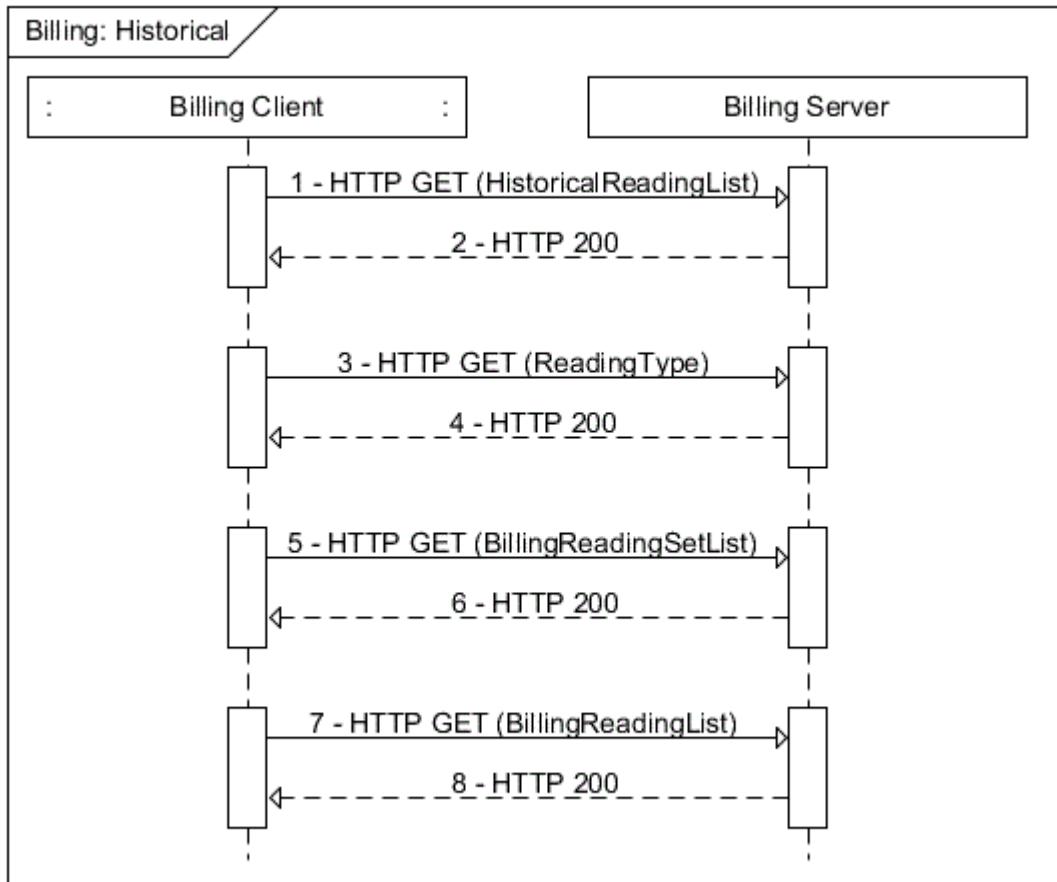
Step	Description
1	<p>Client GETs the CustomerAccount from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /ca/1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
2	<p>Billing server responds with the CustomerAccount.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <CustomerAccount href="/bill/1" xmlns="http://zigbee.org/sep"> <mRID>26d0c9722dd639ab0000e566</mRID> <description/> <currency>840</currency> <customerAccount>981273648</customerAccount> <CustomerAgreementListLink all="1" href="/bill/1/ca"/> <customerName>John Doe</customerName> <pricePowerOfTenMultiplier>-6</pricePowerOfTenMultiplier> <ServiceSupplierLink href="/ss"/> </CustomerAccount></pre>
3	<p>Client GETs the ServiceSupplier from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /ss HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
4	<p>Billing server responds with the ServiceSupplier.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ServiceSupplier href="/ss" xmlns="http://zigbee.org/sep"> <mRID>cac046d1ee2a332a0000e566</mRID> <description>Watts R Us</description> <email>customerservice@wattsRus.com</email> <phone>888.555.1212</phone> <providerID>58726</providerID> <web>www.WattsRus.com</web> </ServiceSupplier></pre>
5	<p>Client GETs the CustomerAgreementList from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /bill/1/ca HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>

Step	Description
6	<p>Billing server responds with the CustomerAgreementList.</p> <p>Server sends the following response:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <CustomerAgreementList all="1" href="/bill/1/ca" results="1" subscribable="0" xmlns="http://zigbee.org/sep"> <CustomerAgreement href="/bill/1/ca/1"> <mRID>65f839fc951345e70000e566</mRID> <description>Electric Service - 4/1/2012</description> <ActiveBillingPeriodListLink all="1" href="/bill/1/ca/1/actbp"/> <BillingPeriodListLink all="1" href="/bill/1/ca/1/bp"/> <HistoricalReadingListLink all="1" href="/bill/1/ca/1/ver"/> <ProjectionReadingListLink all="1" href="/bill/1/ca/1/pro"/> <serviceLocation>Acct. XXX-XXXXX-384 (Elm St.)</serviceLocation> <TariffProfileLink href="/tp/3"/> <UsagePointLink href="/upt/1"/> </CustomerAgreement> </CustomerAgreementList></pre>
7	<p>Client GETs the BillingPeriodList from the Billing Server.</p> <p>Client sends the following request:</p> <pre> GET /bill/1/ca/1/bp?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
8	<p>Billing server responds with the BillingPeriodList.</p> <p>Server sends the following response:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <BillingPeriodList all="2" href="/bill/1/ca/1/bp" results="1" subscribable="1" xmlns="http://zigbee.org/sep"> <BillingPeriod href="/bill/1/ca/1/bp"> <billLastPeriod>140730000</billLastPeriod> <billToDate>83550000</billToDate> <interval> <duration>2419200</duration> <start>1360195200</start> </interval> <statusTimeStamp>1361577600</statusTimeStamp> </BillingPeriod> </BillingPeriodList></pre>

7843 **16.16 Billing: Historical**

7844 This flow diagram describes a billing client device obtaining historical billing readings.



7845

7846

Figure 16-16: Billing Historical

7847 **Table 16-18 POX Example: Billing Historical.**

Step	Description
1	<p>Client GETs the HistoricalReadingList from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /bill/1/ca/1/ver HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
2	<p>Billing server responds with the HistoricalReadingList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <HistoricalReadingList all="1" href="/bill/1/ca/1/ver" results="1"</pre>

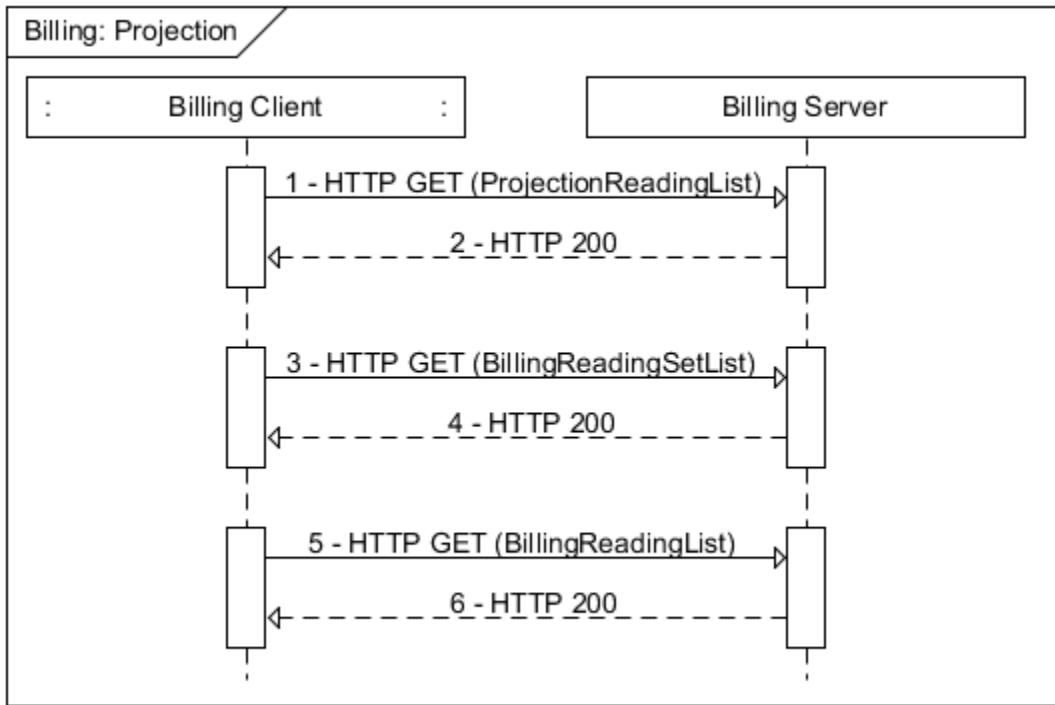
Step	Description
	<pre> xmlns="http://zigbee.org/sep"> <HistoricalReading href="/bill/1/ca/1/ver/1"> <mRID>ce7f99f087b5e3de0000e566</mRID> <description>Hourly usage (kWh)</description> <BillingReadingSetListLink all="180" href="/bill/1/ca/1/ver/1/brs"/> <ReadingTypeLink href="/rt/3"/> </HistoricalReading> </HistoricalReadingList> </pre>
3	<p>Client GETs the ReadingType from the Billing Server.</p> <p>Client sends the following request:</p> <pre> GET /rt/3 HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>
4	<p>Billing server responds with the ReadingType.</p> <p>Server sends the following response:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ReadingType href="/rt/3" xmlns="http://zigbee.org/sep"> <accumulationBehaviour>4</accumulationBehaviour> <commodity>1</commodity> <dataQualifier>12</dataQualifier> <flowDirection>1</flowDirection> <intervalLength>3600</intervalLength> <kind>12</kind> <numberOfConsumptionBlocks>2</numberOfConsumptionBlocks> <numberOfTouTiers>3</numberOfTouTiers> <phase>0</phase> <powerOfTenMultiplier>0</powerOfTenMultiplier> <tieredConsumptionBlocks>false</tieredConsumptionBlocks> <uom>72</uom> </ReadingType> </pre>
5	<p>Client GETs the BillingReadingSetList from the Billing Server.</p> <p>Client sends the following request:</p> <pre> GET /bill/1/ca/1/ver/1/brs?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml </pre>
6	<p>Billing server responds with the BillingReadingSetList.</p> <p>Server sends the following response:</p> <pre> HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <BillingReadingSetList all="180" href="/bill/1/ca/1/ver/1/brs" results="1" subscribable="1" xmlns="http://zigbee.org/sep"> <BillingReadingSet href="/bill/1/ca/1/ver/1/brs/1"> <mRID>82866ecc81c9638a0000e566</mRID> <timePeriod> <duration>86400</duration> <start>1361491200</start> </timePeriod> <BillingReadingListLink all="24" href="/bill/1/ca/1/ver/1/brs/1;br"/> </BillingReadingSet> </BillingReadingSetList> </pre>

Step	Description
7	<p>Client GETs the BillingReadingList from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /bill/1/ca/1/ver/1/brs/1/br?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
8	<p>Billing server responds with the BillingReadingList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <BillingReadingList all="24" href="/bill/1/ca/1/ver/1/brs/1/br" results="1" xmlns="http://zigbee.org/sep"> <BillingReading href="/bill/1/ca/1/ver/1/brs/1/br/1"> <consumptionBlock>0</consumptionBlock> <qualityFlags>01</qualityFlags> <timePeriod> <duration>3600</duration> <start>1361491200</start> </timePeriod> <touTier>3</touTier> <value>38</value> <Charge> <kind>0</kind> <value>429400</value> </Charge> </BillingReading> </BillingReadingList></pre>

7848

7849 **16.17 Billing: Projection**

7850 This flow diagram describes a billing client device obtaining a billing projection.



7851

7852

Figure 16-17: Billing Projection

7853

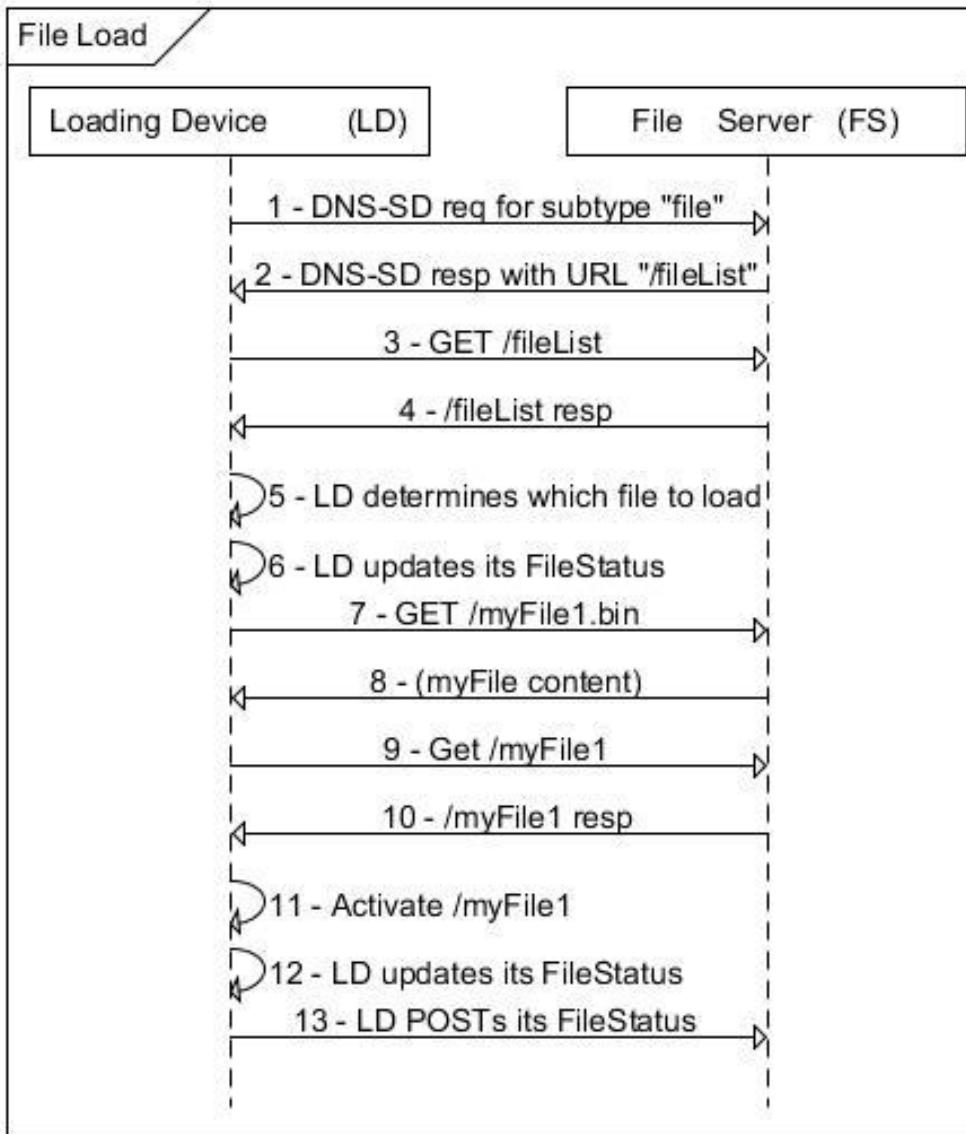
Table 16-19 POX Example: Billing Projection.

Step	Description
1	<p>Client GETs the ProjectionReadingList from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /bill/1/ca/1/pro HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
2	<p>Billing server responds with the ProjectionReadingList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <ProjectionReadingList all="2" href="/bill/1/ca/1/pro" results="1" xmlns="http://zigbee.org/sep"> <ProjectionReading href="/bill/1/ca/1/pro/1"> <mRID>d0b05a2e65144fc0000e566</mRID> <description>Billing Projections</description> <BillingReadingSetListLink all="1" href="/bill/1/ca/1/pro/1/brs"/> </ProjectionReading> </ProjectionReadingList></pre>

Step	Description
3	<p>Client GETs the BillingReadingSetList from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /bill/1/ca/1/pro/1/brs?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
4	<p>Billing server responds with the BillingReadingSetList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <BillingReadingSetList all="1" href="/bill/1/ca/1/pro/1/brs" results="1" subscribable="1" xmlns="http://zigbee.org/sep"> <BillingReadingSet href="/bill/1/ca/1/pro/1/brs/1"> <mRID>7cb8a5b136a9618e0000e566</mRID> <description>Start Consumption Block 2</description> <timePeriod> <duration>2419200</duration> <start>1360195200</start> </timePeriod> <BillingReadingListLink all="1" href="/bill/1/ca/1/pro/1/brs/1/br"/> </BillingReadingSet> </BillingReadingSetList></pre>
5	<p>Client GETs the BillingReadingList from the Billing Server.</p> <p>Client sends the following request:</p> <pre>GET /bill/1/ca/1/pro/1/brs/1/br?l=1 HTTP/1.1 Host: {hostname} Accept: application/sep+xml</pre>
6	<p>Billing server responds with the BillingReadingList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} <BillingReadingList all="24" href="/bill/1/ca/1/ver/1/brs/1/br" results="1" xmlns="http://zigbee.org/sep"> <BillingReading href="/bill/1/ca/1/ver/1/brs/1/br/1"> <consumptionBlock>0</consumptionBlock> <qualityFlags>01</qualityFlags> <timePeriod> <duration>3600</duration> <start>1361491200</start> </timePeriod> <touTier>3</touTier> <value>38</value> <Charge> <kind>0</kind> <value>429400</value> </Charge> </BillingReading> </BillingReadingList></pre>

7854 **16.18 File Loading**

7855 The flow diagram below describes how an LD queries an FS for a list of available files, loads a file from
 7856 the FS, and verifies and activates the loaded file. It also describes how the LD may optionally maintain
 7857 status of the file loading operation and reporting of same to the FS. The flow assumes network joining
 7858 and device registration with service provider have already occurred.



7859

7860

Figure 16-18: File Load - FlowDiagram

7861 Step descriptions are listed below.

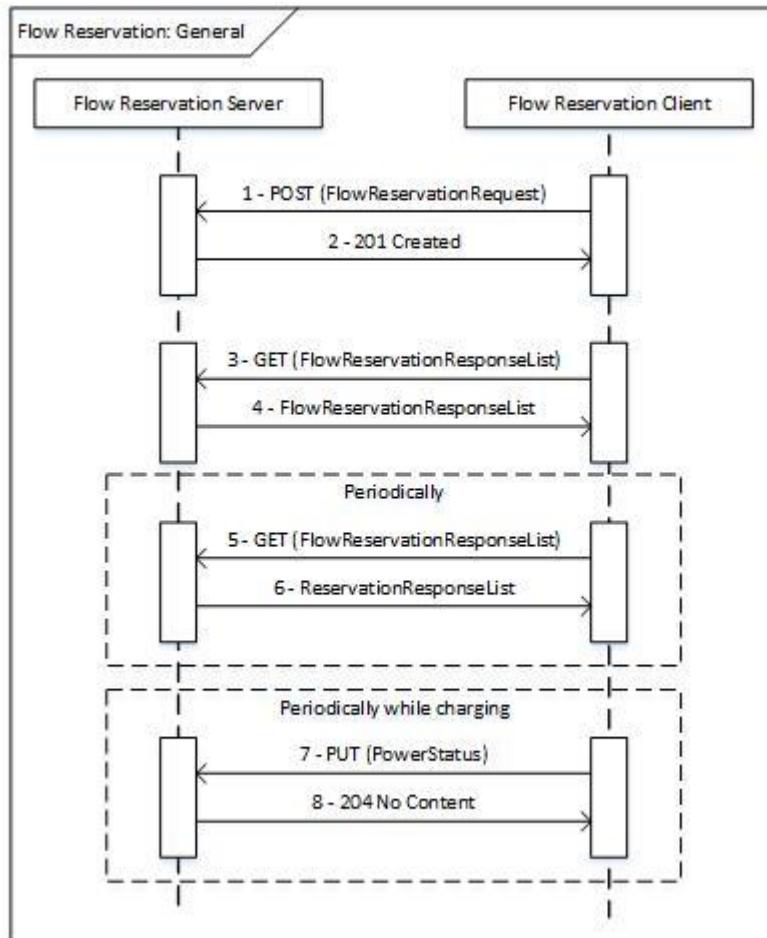
7862

Table 16-20 File Load: Flow Description

Step	Description
1	The LD issues discovery requests to locate available FS (DNS-SD subtype "file").
2	The LD has discovered the FS and obtained the URL of its fileList.
3	<p>The LD queries the fileList to determine if there are any available files to be downloaded to the LD. An example fileList query:</p> <pre>GET http://host1/fileList?s=0&l=5&type=0x0000&mfId=37244&mfModel=123abc&mfVer=23.47.102 HTTP/1.1 Host: host1</pre> <p>This query directs the FS to return the list of firmware files (type) from manufacturer 37244 (IANA PEN) LD model "123abc" whose version number is greater than (newer than) 23.47.102. As the LFDI and mfHwVer were omitted, they implicitly match any value that may have been specified in the File resources.</p>
4	<p>The FS responds to the LD with the fileList satisfying the query parameters.</p> <p>An example FS response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: ... <FileList all="2" href="http://host1/fileList" results="2" xmlns="http://zigbee.org/sep"> <File href="http://host1/myFile1"> <fileURI>http://host1/myfile1.bin</fileURI> <mfID>37244</mfID> <mfModel>123abc</mfModel> <mfVer>23.48.1</mfVer> <size>128000</size> <type>00</type> </File> <File href="http://host1/myFile2"> <fileURI>http://host1/myfile2.bin</fileURI> <mfID>37244</mfID> <mfModel>123abc</mfModel> <mfVer>23.47.103</mfVer> <size>136000</size> <type>00</type> </File> </FileList></pre> <p>The FS reports that it has two files which match the initial query for firmware/manufacturer PEN 37244, model "123abc", and are newer than version 23.47.102. The first entry in the list (mfVer 23.48.1) is the latest version.</p> <p>Note: no activation time was provided.</p>
5	The LD examines the results from the fileList query and determines which, if any, of the File resources should be loaded. The LD selects the latest file available /myFile1 with version 12.48.1
6	The LD updates its FileStatus resource.
7-8	The LD loads /myFile1 with version 12.48.1. Note: this exchange will often be accomplished with multiple HTTP(S) request/response (using the Range and Content-Range entity headers).

Step	Description
9	<p>Recall that no activation time was included in the original /myFile1 obtained by the LD from the FileList. Whenever a LD loads a File with an unspecified activateTime, the LD will continue to poll to acquire a file activateTime (interval and retry discussed above).</p> <p>The LD will periodically issue the following request:</p> <pre>GET http://host1/myFile1 HTTP/1.1 Host: host1</pre>
10	<p>The FS responds with /myFile1.</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: ... <File href="http://host1/myFile1" xmlns="http://zigbee.org/sep"> <activateTime>3763784682</activateTime> <fileURI>http://host1/myfile1.bin</fileURI> <mfID>37244</mfID> <mfModel>123abc</mfModel> <mfVer>23.48.1</mfVer> <size>128000</size> <type>00</type> </File></pre> <p>Note that, this time, an activateTime is provided within /myFile1. If there is not an activateTime contained in /myFile1, the LD continues to poll for activateTime.</p>
11	The LD waits until the activation time specified for /myFile1 is reached, then places the file into the activated state. In the case of a firmware file, the file is now the running image.
12	The LD again updates its FileStatus resource.
13	<p>The LD PUTs its FileStatus resource to a remote EndDevice server.</p> <p>In this example, the EndDevice is hosted at the FS. Thus the FS is provided with a final status of the LD load operation.</p>

7863 16.19 Flow Reservation: General



7864

7865

Figure 16-19: Flow Reservation: General

7866 The following is a summary of the example:

7867 Step 1 - Client (PEV) creates a FlowReservationRequest at 5:00 PM for charging between midnight and
7868 8:00 AM, 12 kWh energy requested at a power level of 7 kW, 7371 seconds duration requested.7869 • Subsequently, the Server creates a FlowReservationResponse with a charge interval between
7870 1:00 AM and 5:20 AM at 3 kW.

7871 Step 3 - Client requests the FlowReservationResponseList to find the response matching the request.

7872 Step 4 – Server responds with the FlowReservationResponseList.

7873 Step 5 - Client periodically requests the FlowReservationResponse to look for changes.

7874 Step 7 - Client updates PowerStatus periodically during charging.

7875 Note: In most cases, registration is required to obtain access to request flow reservations.

7876

Table 16-21 POX Example: Flow Reservation - General.

Step	Description
1	<p>Client POSTs a FlowReservationRequest to the Flow Reservation Server at 9/22/2013 5:00 PM.</p> <p>Client sends the following request:</p> <pre>POST /edev/3/frq HTTP/1.1 Host: {hostname} <FlowReservationRequest xmlns="http://zigbee.org/sep"> <mRID>68512866203db3b10000e566</mRID> <description>Charge between 12AM and 8AM</description> <!-- 9/22/2013 5:00 PM GMT --> <creationTime>1379869200</creationTime> <!-- 6171sec charging + 1200sec conditioning --> <durationRequested>7371</durationRequested> <energyRequested> <!-- 12 kWh --> <multiplier>3</multiplier> <value>12</value> </energyRequested> <intervalRequested> <!-- 8 hours --> <duration>28800</duration> <!-- 9/23/2013 12:00 AM GMT --> <start>1379894400</start> </intervalRequested> <powerRequested> <!-- 7 kW --> <multiplier>3</multiplier> <value>7</value> </powerRequested> <RequestStatus> <dateTime>1379869200</dateTime> <!-- Requested --> <requestStatus>0</requestStatus> </RequestStatus> </FlowReservationRequest></pre>
2	<p>Flow Reservation server responds with the FlowReservationRequest location.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 201 Created Location: /edev/3/frq/1</pre>
3	<p>Client GETs the FlowReservationResponseList from the Flow Reservation Server to look for a response to the request.</p> <p>Client sends the following request:</p> <pre>GET /edev/3/frp HTTP/1.1 Host: {hostname}</pre>
4	<p>Flow Reservation server responds with the FlowReservationResponseList</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sepxml</pre>

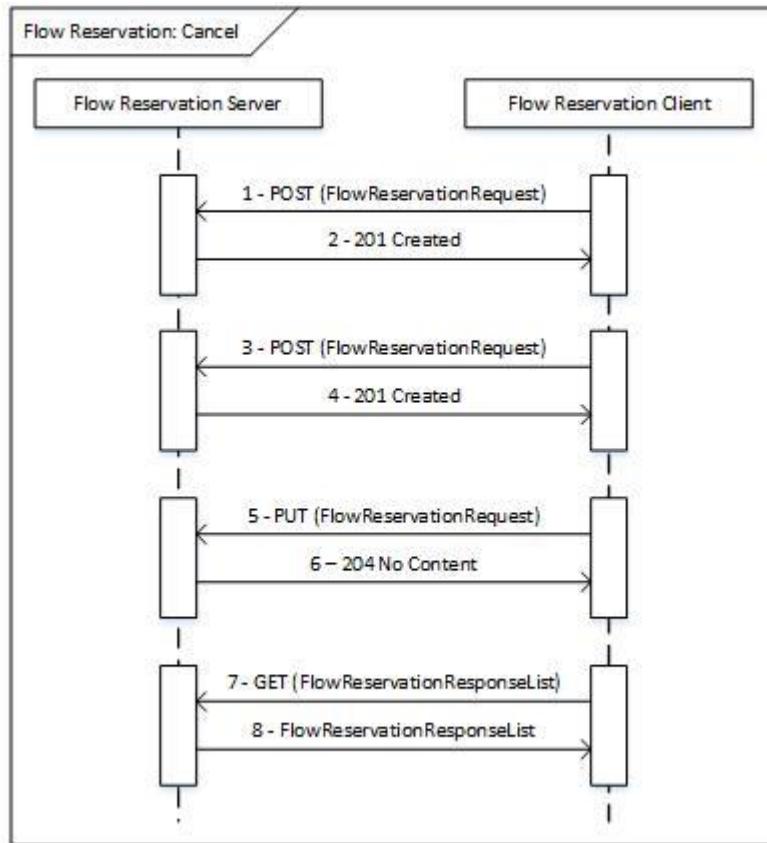
Step	Description
	<pre> <FlowReservationResponseList all="1" href="/edev/3/frp" results="1" subscribable="1" xmlns="http://zigbee.org/sep"> <FlowReservationResponse href="/edev/3/frp/1" subscribable="1"> <mRID>f8afa6fde40db98d0000ea75</mRID> <description>Charge between 1AM and 5:20AM</description> <!-- 9/22/2013 5:01 PM GMT --> <creationTime>1379869260</creationTime> <EventStatus> <!-- Scheduled --> <currentStatus>0</currentStatus> <dateTime>1379869260</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <!-- 4 hours 20 minutes --> <duration>15600</duration> <!-- 9/23/2013 1:00 AM GMT --> <start>1379898000</start> </interval> <energyAvailable> <multiplier>0</multiplier> <value>12000</value> </energyAvailable> <powerAvailable> <multiplier>0</multiplier> <value>3000</value> </powerAvailable> <subject>68512866203db3b10000e566</subject> </FlowReservationResponse> </FlowReservationResponseList> </pre>
5	<p>Client GETs the FlowReservationResponseList periodically (or subscribes) from the Flow Reservation Server.</p> <p>Client sends the following request:</p> <pre>GET /edev/3/frp HTTP/1.1 Host: {hostname}</pre>
6	<p>Flow Reservation server responds with the FlowReservationResponseList.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml <FlowReservationResponseList all="1" href="/edev/3/frp" results="1" subscribable="1" xmlns="http://zigbee.org/sep"> <FlowReservationResponse href="/edev/3/frp/1" subscribable="1"> <mRID>f8afa6fde40db98d0000ea75</mRID> <description>Charge between 1AM and 5:20AM</description> <!-- 9/22/2013 5:01 PM GMT --> <creationTime>1379869260</creationTime> <EventStatus> <!-- Scheduled --> <currentStatus>0</currentStatus> <dateTime>1379869260</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <!-- 4 hours 20 minutes --> <duration>15600</duration> <!-- 9/23/2013 1:00 AM GMT --> <start>1379898000</start> </interval> </FlowReservationResponse> </FlowReservationResponseList></pre>

Step	Description
	<pre> </interval> <energyAvailable> <multiplier>0</multiplier> <value>12000</value> </energyAvailable> <powerAvailable> <multiplier>0</multiplier> <value>3000</value> </powerAvailable> <subject>68512866203db3b10000e566</subject> </FlowReservationResponse> </FlowReservationResponseList></pre>
7	<p>From 1:00 AM to 5:20 AM while charging the client periodically updates its PowerStatus.</p> <p>Client sends the following request:</p> <pre> PUT /edev/3/ps HTTP/1.1 Host: {hostname} <PowerStatus xmlns="http://zigbee.org/sep"> <!-- more than LowChargeThreshold remaining --> <batteryStatus>1</batteryStatus> <!-- 9/23/2013 3:00 AM GMT --> <changedTime>1379905200</changedTime> <!-- mains --> <currentPowerSource>1</currentPowerSource> <estimatedChargeRemaining>7000</estimatedChargeRemaining> <PEVInfo> <chargingPowerNow> <multiplier>0</multiplier> <value>3000</value> </chargingPowerNow> <energyRequestNow> <multiplier>0</multiplier> <value>6100</value> </energyRequestNow> <maxForwardPower> <multiplier>3</multiplier> <value>7</value> </maxForwardPower> <!-- 3600sec * 6100Wh/7000W + 1200sec conditioning --> <minimumChargingDuration>4337</minimumChargingDuration> <targetStateOfCharge>10000</targetStateOfCharge> <!-- 9/23/2013 8:00 AM GMT --> <timeChargeIsNeeded>1379923200</timeChargeIsNeeded> <timeChargingStatusPEV>1379905200</timeChargingStatusPEV> </PEVInfo> </PowerStatus></pre>
8	<p>Flow Reservation server responds:</p> <p>HTTP/1.1 204 No Content</p>

7877

7878

7879 16.20 Flow Reservation: Cancel



7880

7881 **Figure 16-20: Flow Reservation: Cancel**

7882 The following is a summary of the example:

7883 Step 1 - Client (PEV) creates a FlowReservationRequest at 5:00 PM for charging between midnight and
7884 8:00 AM.

- 7885 • Subsequently, the Server creates a FlowReservationResponse with a charge interval between
-
- 7886 1:00 AM and 5:20 AM at 3 kW.

7887 Step 3 – At 1:00 AM the Client wants to change the time the charging is needed and therefore creates a
7888 new FlowReservationRequest for charging between 1:00 AM and 5:00 AM.

- 7889 • Subsequently, the Server creates a FlowReservationResponse with a charge interval between
-
- 7890 1:02 AM and 4:22 AM at 4 kW.

7891 Step 5 - Client cancels the original FlowReservationRequest.

- 7892 • Subsequently, the Server cancels the original FlowReservationResponse.

7893 Step 7 - Client requests the FlowReservationResponseList.

7894 Step 8 – Server responds with the FlowReservationResponseList, the first FlowReservationResponse is
7895 canceled and the second FlowReservationResponse is Active.

7896

Table 16-22 POX Example: Flow Reservation - Cancel

Step	Description
1	<p>Client POSTs a FlowReservationRequest to the Flow Reservation Server at 9/22/2013 5:00 PM.</p> <p>Client sends the following request:</p> <pre> POST /edev/3/frq HTTP/1.1 Host: {hostname} <FlowReservationRequest xmlns="http://zigbee.org/sep"> <mRID>68512866203db3b10000e566</mRID> <description>Charge between 12AM and 8AM</description> <!-- 9/22/2013 5:00 PM GMT --> <creationTime>1379869200</creationTime> <!-- 6171sec charging + 1200sec conditioning --> <durationRequested>7371</durationRequested> <energyRequested> <!-- 12 kWh --> <multiplier>3</multiplier> <value>12</value> </energyRequested> <intervalRequested> <!-- 8 hours --> <duration>28800</duration> <!-- 9/23/2013 12:00 AM GMT --> <start>1379894400</start> </intervalRequested> <powerRequested> <!-- 7 kW --> <multiplier>3</multiplier> <value>7</value> </powerRequested> <RequestStatus> <dateTime>1379869200</dateTime> <!-- Requested --> <requestStatus>0</requestStatus> </RequestStatus> </FlowReservationRequest></pre>
2	<p>Flow Reservation server responds with the FlowReservationRequest location.</p> <p>Server sends the following response:</p> <pre> HTTP/1.1 201 Created Location: /edev/3/frq/1 </pre>
3	<p>Due to a change in when the vehicle is needed, at 9/23/2013 1:00 AM the client creates a second FlowReservationRequest for charging between 1:00 AM and 5:00 AM.</p> <p>Client sends the following request:</p> <pre> POST /edev/3/frq HTTP/1.1 Host: {hostname} <FlowReservationRequest xmlns="http://zigbee.org/sep"> <mRID>68512866203db3b10000e577</mRID> <description>Charge between 1AM and 5AM</description> <!-- 9/23/2013 1:00 AM GMT --> <creationTime>1379898000</creationTime> <!-- 6171sec charging + 1200sec conditioning --> <durationRequested>7371</durationRequested> <energyRequested> <!-- 12 kWh --> </energyRequested></pre>

Step	Description
	<pre> <multiplier>3</multiplier> <value>12</value> </energyRequested> <intervalRequested> <!-- 4 hours --> <duration>14400</duration> <!-- 9/23/2013 1:00 AM GMT --> <start>1379898000</start> </intervalRequested> <powerRequested> <!-- 7 kW --> <multiplier>3</multiplier> <value>7</value> </powerRequested> <RequestStatus> <dateTime>1379898000</dateTime> <!-- Requested --> <requestStatus>0</requestStatus> </RequestStatus> </FlowReservationRequest> </pre>
4	<p>Flow Reservation server responds with the FlowReservationRequest location.</p> <p>Server sends the following response:</p> <p>HTTP/1.1 201 Created Location: /edev/3/frq/2</p>
5	<p>Client PUTs a canceled status to the first FlowReservationRequest.</p> <p>Client sends the following request:</p> <pre> PUT /edev/3/frq/1 HTTP/1.1 Host: {hostname} <FlowReservationRequest xmlns="http://zigbee.org/sep"> <mRID>68512866203db3b10000e566</mRID> <description>Charge between 12AM and 8AM</description> <!-- 9/22/2013 5:00 PM --> <creationTime>1379869200</creationTime> <durationRequested>7371</durationRequested> <energyRequested> <multiplier>3</multiplier> <value>12</value> </energyRequested> <intervalRequested> <duration>28800</duration> <start>1379894400</start> </intervalRequested> <powerRequested> <multiplier>3</multiplier> <value>7</value> </powerRequested> <RequestStatus> <!-- 9/23/2013 1:01 AM GMT --> <dateTime>1379898060</dateTime> <!-- Canceled --> <requestStatus>1</requestStatus> </RequestStatus> </FlowReservationRequest> </pre>
6	<p>Flow Reservation server sends the following response:</p> <p>HTTP/1.1 204 No Content</p>

Step	Description
7	<p>Client GETs the FlowReservationResponseList periodically (or subscribes) from the Flow Reservation Server.</p> <p>Client sends the following request:</p> <pre>GET /edev/3/frp?l=2 HTTP/1.1 Host: {hostname}</pre>
8	<p>Flow Reservation server responds with the FlowReservationResponseList. The first FlowReservationResponse is Canceled, the second is Active.</p> <p>Server sends the following response:</p> <pre>HTTP/1.1 200 OK Content-Type: application/sep+xml <FlowReservationResponseList all="2" href="/edev/3/frp" results="2" subscribable="1" xmlns="http://zigbee.org/sep"> <FlowReservationResponse href="/edev/3/frp/1" subscribable="1"> <mRID>f8afa6fde40db98d0000ea75</mRID> <description>Charge between 1AM and 5:20AM</description> <!-- 9/22/2013 5:01 AM GMT --> <creationTime>1379869260</creationTime> <EventStatus> <!-- Canceled --> <currentStatus>2</currentStatus> <dateTime>1379898060</dateTime> <potentiallySuperseded>true</potentiallySuperseded> </EventStatus> <interval> <!-- 4 hours 20 minutes --> <duration>15600</duration> <!-- 9/23/2013 1:00 AM GMT --> <start>1379898000</start> </interval> <energyAvailable> <multiplier>0</multiplier> <value>12000</value> </energyAvailable> <powerAvailable> <multiplier>0</multiplier> <value>3000</value> </powerAvailable> <subject>68512866203db3b10000e566</subject> </FlowReservationResponse> <FlowReservationResponse href="/edev/3/frp/1" subscribable="1"> <mRID>f8afa6fde40db98d0000ea76</mRID> <description>Charge between 1:02AM and 4:22AM</description> <!-- 9/23/2013 1:02 AM GMT --> <creationTime>1379898120</creationTime> <EventStatus> <!-- Active --> <currentStatus>1</currentStatus> <dateTime>1379898120</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <!-- 3 hours 20 minutes --> <duration>12000</duration> <!-- 9/23/2013 1:02 AM GMT --> <start>1379898120</start> </interval> </FlowReservationResponse> </FlowReservationResponseList></pre>

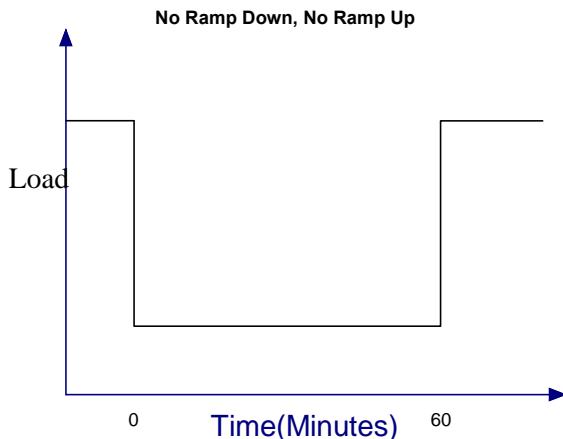
Step	Description
7897	<pre> </interval> <energyAvailable> <multiplier>0</multiplier> <value>12000</value> </energyAvailable> <powerAvailable> <multiplier>0</multiplier> <value>4000</value> </powerAvailable> <subject>68512866203db3b10000e577</subject> </FlowReservationResponse> </FlowReservationResponseList></pre>

7897

7898 16.21 Event Randomization

7899 The following are examples of how to set the randomization parameters to accomplish different
 7900 randomization strategies. The attributes of an event that define its behavior in time are, the "interval" that
 7901 defines the "start" and "duration" of the event, "randomizeStart" and "randomizeDuration". The later two
 7902 controlling the randomization behavior. For these examples, we will assume we are looking at DRLC
 7903 events and that the events are causing reductions in load over a large population. The graphs are
 7904 indicating the aggregated load for the entire population for a single event.

7905 16.21.1 Simple Event



7906

7907 **Figure 16-21: Simple Event - No Ramp Up - No Ramp Down**

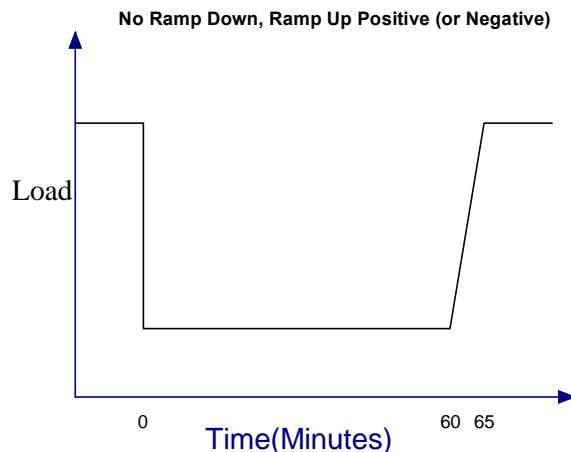
7908 Note: This use case is NOT targeted for large population control.

7909 **Table 16-23 Ramp Value.**

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	0
randomizeDuration	0

7910 16.21.2 Event with Positive Randomized Duration

7911



7912

7913 **Figure 16-22: Event with Positive Randomization Duration**

7914

7915 **Table 16-24 Event Ramp Time.**

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	0
randomizeDuration	300

7916 16.21.3 Event with Positive Randomized Start

7917

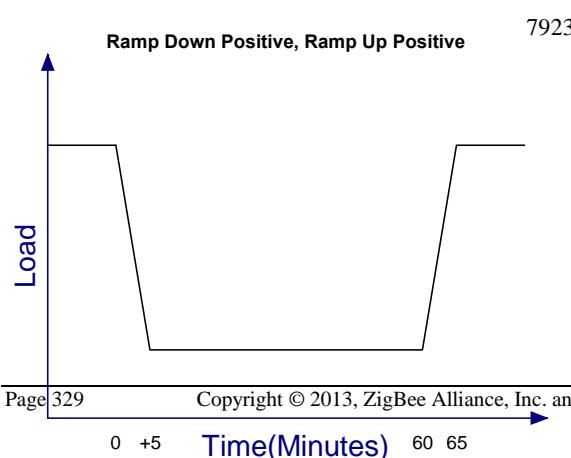
7918

7919

7920

7921

7922

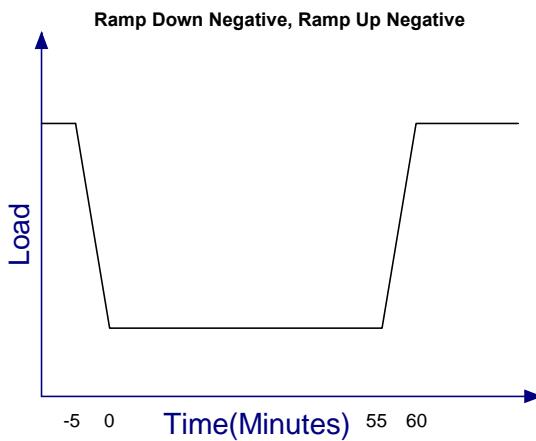


7923

7924

7925 **Figure 16-23: Event with Positive Randomization Start**7926 **Table 16-25 Event Start Time.**

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	300
randomizeDuration	0

7927 **16.21.4 Event with Negative Randomized Start**

7928

7929 **Figure 16-24: Event with Negative Randomized Start**

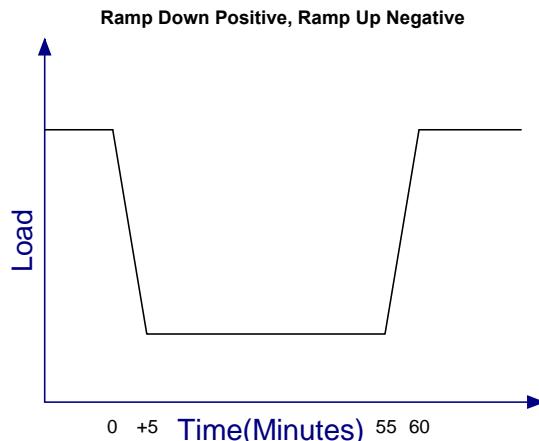
7930

7931

7932 **Table 16-26 Event Start Time.**

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	-300
randomizeDuration	0

7933 16.21.5 Event with Positive Randomization and Finishing in one Hour



7934

Figure 16-25: Positive Randomization in an Hour

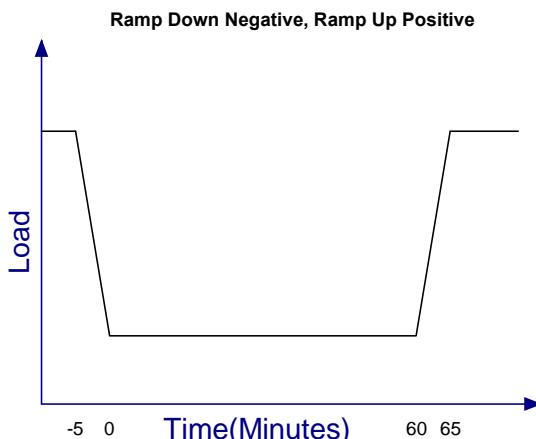
7935

Table 16-27 Event Start Time.

Attribute	Value (seconds)
start	0
duration	3300
randomizeStart	300
randomizeDuration	0

7937

16.21.6 Event with Negative Randomized Start and at Least One Hour Duration



7938

Figure 16-26: Event with Negative Start in One Hour

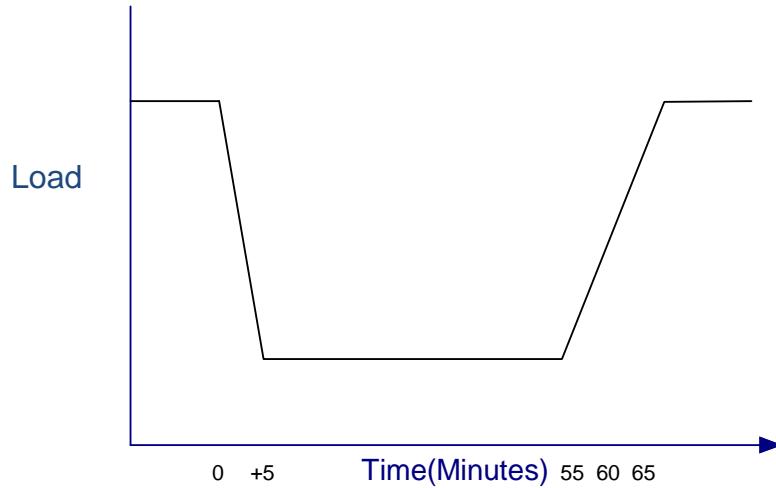
7939

Table 16-28 Event Start Time.

Attribute	Value (seconds)
start	0
duration	3900

randomizeStart	-30041
randomizeDuration	0

7942

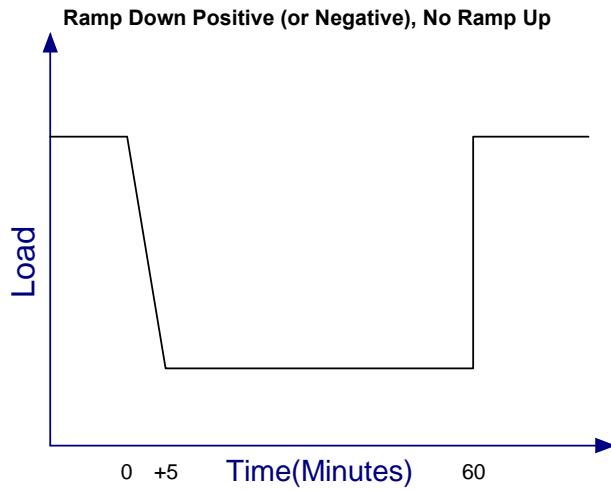
7943 16.21.7 **Event with Randomized Start and Long Ramp Up****Ramp Down Positive, Random Duration**

7944

7945 **Figure 16-27: Randomized Start and Long Ramp Up**7946 **Table 16-29 Event Start Time.**

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	300
randomizeDuration	300

7947

7948 16.21.8 **Event with Positive Randomized Start and Fixed End Time**

7949

Figure 16-28: Randomized Start and Fixed End Time

7950 Note: This scenario is NOT possible with the current set of parameters. For grid stability "snap" back on
7951 is not a desirable use case.

7953 17 Appendix D – Guidelines [INFORMATIVE]**7954 17.1 Pricing Implementation Guidelines**

7955 This section discusses guidelines for implementing common service provider tariff rate designs and
7956 enabling HAN devices to match a TariffProfile.RateComponent.ReadingType with a meaningful
7957 Metering value in cases where the Pricing and Metering servers may not be coordinated or match.

7958 This section assumes readers are familiar with the Pricing and Metering function set text in early sections.
7959 Readers may also wish to consult the WADL and schema [ZB 13-0201] for more detailed information.

7960 17.1.1 Implementing Common Tariff Designs

7961 This section is intended to provide guidance to service providers and HAN server implementers on how to
7962 use the Pricing function set and its design flexibility in a way that encourages standard uses and practices.
7963 High level description of the resources and attribute values are provided here; for detailed XML examples
7964 of the pricing function set, see Section 16.14. The following four tariff rate designs are discussed further
7965 in the text below:

- 7966 • Flat rate
- 7967 • Time-of-use tiers
- 7968 • Consumption-based blocks
- 7969 • Time-of-use tiers with consumption-based blocks

7970 All four designs will assume that pricing information is only provided for the forward direction
7971 (commodity delivered), and is applicable for any flow rate (demand). This corresponds to a TariffProfile
7972 with a single RateComponent in its RateComponentList. This *RateComponent* would have the
7973 flowRateStartLimit and flowRateEndLimit attributes elided, and would link to a ReadingType with the
7974 flowDirection attribute set to "1".

7975 17.1.2 Flat Rate Design

7976 The flat rate design assumes a constant price for a commodity over one or many billing periods. With no
7977 price differentiation based on the time of day or on the amount of the commodity already consumed, the
7978 flat-rate design requires only a single *touTier* and a single *consumptionBlock*. Therefore the tariff's
7979 RateComponent must link to a ReadingType with the following attributes:

- 7980 • numberOfWorkingBlocks = 1
- 7981 • numberofTouTiers = 1

7982 Per [ZB 13-0201], the RateComponent links to a TimeTariffIntervalList. The flat rate design only
7983 requires a single TimeTariffInterval in this list, though information for future seasons could be
7984 communicated by providing additional TimeTariffIntervals in the list (only one TimeTariffInterval can be
7985 active at a given time). Any TimeTariffInterval instance provided under a flat rate design must have the
7986 *touTier* attribute set to "1".

7987 Again per [ZB 13-0201], TimeTariffInterval links to a ConsumptionTariffIntervalList. For the flat rate
7988 design there must only be one ConsumptionTariffInterval in the list. This ConsumptionTariffInterval
7989 defines the consumption price per commodity unit. The *startValue* attribute in this
7990 ConsumptionTariffInterval must be "0" and the *consumptionBlock* attribute must be "1".

7991 17.1.3 **Time-of-Use Rate Design**

7992 Most TOU rate designs consist of a repeating series of time-differentiated rates. These can vary over the
 7993 course of a day, and the series may be different on weekdays and weekends. Pricing servers may be
 7994 implemented and configured with functionality that creates repeating TimeTariffIntervals based on an
 7995 internal schedule in order to minimize backhaul network traffic, but this is out of scope for SEP 2. For the
 7996 HAN, such a Pricing server would supply a steady stream of pricing "events" with a defined
 7997 commencement and duration. Pricing service providers and their servers are encouraged to provide
 7998 enough pricing information to cover the next 24-hour period since most consumer devices operate over
 7999 this time scale. Pricing service providers are also encouraged to update their Pricing servers with the next
 8000 sequential TimeTariffInterval instance at least four hours before its Effective Start Time.

8001 For this TOU rate example, assume a daily three-tier (plus one Critical Peak Pricing (CPP) tier) rate
 8002 design with the following characteristics as shown in Table 17-1:

8003 **Table 17-1 TOU Rate.**

Time Period	Description	Price	Tier
Midnight – 8 AM	Off-peak	\$0.10	1
8 AM – 10 AM	Mid-peak	\$0.20	2
10 AM – 6 PM	On-peak	\$0.40	3
6 PM – Midnight	Mid-peak	\$0.20	2
Intermittent	CPP	\$0.70	4

8004 Tier numbers are ordered by price from lowest to highest (though note that the *price* attribute itself will be
 8005 found in the ConsumptionTariffInterval resource linked to by a TimeTariffInterval, not in the
 8006 TimeTariffInterval itself); this is mandatory in Smart Energy 2.0.

8007 The TOU design provides price differentiation based on the time of day. However, as with the flat rate
 8008 design, there is no price differentiation based on the amount of the commodity already consumed. In SEP
 8009 2 terms, this means that the TOU design requires two or more *touTiers*, but must provide only a single
 8010 *consumptionBlock*. Using the above TOU example, the tariff's RateComponent must link to a
 8011 ReadingType with the following attributes:

- *numberOfConsumptionBlocks* = 1
- *numberOfTouTiers* = 4

8014 Continuing the above example, consider a pricing server configured as follows:

- Server provides 48 hours of price information.
- Price information is updated every midnight.
- The local time is 9 AM on July 16, 2012.
- There are no CPP events scheduled for the next 48 hours.

8019 In this scenario, the server's TimeTariffIntervalList will contain 8 TimeTariffIntervals: one expired event,
 8020 one active event, and six scheduled events. Each TimeTariffInterval will link to a
 8021 ConsumptionTariffIntervalList, where each list, under a TOU-only design, contains one
 8022 ConsumptionTariffInterval. As with the flat rate design, each ConsumptionTariffInterval must have a
 8023 *consumptionBlock* attribute of "1" and a *startValue* attribute of "0".

8024 17.1.4 **Consumption-based Block Rate Design**

8025 In some jurisdictions (e.g., California, British Columbia, the UK), consumption-based block rate designs
 8026 have been put in place to incentivize efficient consumption of a commodity with prices that ratchet up
 8027 over the course of the billing period based on the amount of the commodity consumed. In other scenarios,
 8028 typically for commercial and industrial (C&I) customers, the price charged per commodity unit consumed
 8029 falls after some usage threshold.

8030 For this consumption-based rate design, assume a five-tier design with the following characteristics in
 8031 Table 17-2:

8032 **Table 17-2 Consumption-based Rates.**

Block	Consumption startValue	Price
1	0	\$0.12
2	150	\$0.14
3	250	\$0.23
4	300	\$0.27
5	350	\$0.30

8033 Block numbers are ordered by *startValue* from lowest to highest; this is mandatory in SEP 2. While the
 8034 associated prices in this example also increase with each block number, this is not mandated by SEP 2.
 8035 The above example corresponds to a *ReadingType* with the following attributes:

- 8036 • *numberOfConsumptionBlocks* = 5
- 8037 • *numberOfTouTiers* = 1

8038 As with the flat rate design, a server using the block rate design may only have one *TimeTariffInterval* in
 8039 the *TimeTariffIntervalList* linked to from *RateComponent*. That *TimeTariffInterval* could correspond to
 8040 an entire billing period or longer. However, unlike the flat rate design, that one *TimeTariffInterval* will
 8041 link to a *ConsumptionTariffIntervalList* containing multiple (5, in this example)
 8042 *ConsumptionTariffInterval* resources.

8043 Note that under a block rate design, as opposed to flat or TOU rate designs, a pricing client cannot
 8044 determine the active price solely from the pricing server. The client will also have to query the
 8045 corresponding metering server to determine which consumption block is active.

8046 17.1.5 **Combined Time-of-Use and Consumption-based Block Rate Design**

8047 Yet other jurisdictions (e.g., California), have combined time-of-use and consumption-based rate designs
 8048 to incentivize efficient consumption as well as shifting that consumption to times of lower system
 8049 demand.

8050 For this TOU with consumption-based rate design, assume a tariff structure that combines the previous
 8051 two examples as shown in Table 17-3:

8052 **Table 17-3 Tariff Structure.**

Time	Tier	Block	Consumption startValue	Description	Price
Midnight – 8 AM	1	1	0	Off-peak	\$0.22

Time	Tier	Block	Consumption startValue	Description	Price
8 AM – 10 AM	2	2	150	Mid-peak	\$0.24
		3	250		\$0.33
		4	300		\$0.37
		5	350		\$0.40
10 AM – 6 PM	3	1	0	On-peak	\$0.32
		2	150		\$0.34
		3	250		\$0.43
		4	300		\$0.47
		5	350		\$0.50
6 PM - Midnight	2	1	0	Mid-peak	\$0.52
		2	150		\$0.54
		3	250		\$0.73
		4	300		\$0.77
		5	350		\$0.80
Intermittent	4	1	0	CPP	\$0.32
		2	150		\$0.34
		3	250		\$0.43
		4	300		\$0.47
		5	350		\$0.50

8053 Using the above TOU+Block example, the tariff's RateComponent must link to a ReadingType with the
 8054 following attributes:

- 8055 • numberOfConsumptionBlocks = 5
 8056 • numberOfTouTiers = 4

8057 Continuing the above example, consider a pricing server configured as follows:

- 8058 • Server provides 48 hours of price information.
 8059 • Price information is updated every midnight.
 8060 • The local time is 9 AM on July 16, 2012.
 8061 • There is a CPP event scheduled for 1 PM – 3 PM on July 16, 2012.

8062 In this scenario, the server's TimeTariffIntervalList will contain 10 TimeTariffIntervals:

- 8063 1) One expired event (Tier 1, present day 00:00:00 – 08:00:00)

- 8064 2) One active event (Tier 2, present day 08:00:00 – 10:00:00)
8065 3) Eight scheduled events, corresponding to:
8066 1) Tier 3, present day 10:00:00 – 13:00:00
8067 2) Tier 4 (CPP), present day 13:00:00 – 15:00:00
8068 3) Tier 3, present day 15:00:00 – 18:00:00
8069 4) Tier 2, present day 18:00:00 – next day 00:00:00
8070 5) Tier 1, next day 00:00:00 – 08:00:00
8071 6) Tier 2, next day 08:00:00 – 10:00:00
8072 7) Tier 3, next day 10:00:00 – 18:00:00
8073 8) Tier 2, next day 18:00:00 – day after next 00:00:00
- 8074 Each TimeTariffInterval links to its own ConsumptionTariffIntervalList, and each
8075 ConsumptionTariffIntervalList will contain 5 ConsumptionTariffInterval resources with
8076 *consumptionBlock*, *startValue*, and *price* attributes set per the above TOU+Block rate.
- 8077 As with the block rate design, under a TOU+Block design a pricing client cannot determine the active
8078 price solely from the pricing server; the client must determine the active block from the associated
8079 metering server.
- 8080 **17.1.6 Coordinated and Uncoordinated Pricing and Metering Servers**
- 8081 As with all function set servers in SEP 2, Pricing and Metering servers may be hosted by different service
8082 providers, served by the same provider on separate hosts, or coordinated or uncoordinated depending on
8083 the rules of the jurisdiction. This is problematic for certain tariff designs, particularly for consumption-
8084 based tariffs, which depend on usage accumulated during a given billing period in order to be able to
8085 provide accurate Pricing information. For example, in Texas, the Retail Energy Provider (REP) owns the
8086 customer relationship and is responsible for serving Pricing information, but the Transmission
8087 Distribution Service Provider (TDSP) owns the meter and provides one standard meter configuration for
8088 all users, regardless of the REP, which may change depending on the customer's preferences. Another use
8089 case is a customer who may not have a smart meter but installs his own and links his HAN to his
8090 electricity service provider's Pricing server on its website.
- 8091 This section recommends mitigation strategies and coping mechanisms to help Pricing and Metering
8092 clients provide users with actionable information to make informed decisions in the event that the
8093 ReadingType instance referenced by the Pricing server does not match any of the Metering server
8094 implementations available in the HAN. Devices that follow Pricing servers primarily for price
8095 responsiveness (e.g., smart appliances) do not need to follow these recommendations because the
8096 *numberOfTouTiers* and *touTier* attributes in the pricing server's ReadingType and TimeTariffInterval
8097 resources provide the guidance needed to optimize operational parameters.
- 8098 The following rules are recommended for clients of Pricing and Metering resources and which aim to
8099 present a user with cost information about their usage or determine a nominal price for energy. These
8100 rules presume a client has performed service discovery and identified a suitable Metering server for its
8101 application (e.g., the premises aggregation point, PEV sub-meter). The first three rules assume that the
8102 Pricing and Metering servers are coordinated. Rules four and five are meant to guide implementations in
8103 scenarios where the Pricing and Metering servers are uncoordinated.

- 8104 1) Metering servers SHOULD, at a minimum, serve a ReadingType instance for commodity
8105 delivered summation. This provides a functionality baseline that all devices can expect to be
8106 available in a HAN.
- 8107 2) If Rule #1 is not satisfied, and if the ReadingTypeLink URI in the Pricing server's
8108 RateComponent object and the Metering server's MeterReading object are the same, then they are
8109 a match.
- 8110 3) If Rule #2 is not satisfied, then the client SHOULD refer to the MeterReading instances' mRID
8111 attributes. If the Pricing and Metering servers' MeterReading.mRID attributes are the same, then
8112 they are a match.
- 8113 4) If Rule #3 is not satisfied, then the client should compare values in order to find the best fit.
 - 8114 a) The following attributes SHALL be exact matches in both servers' ReadingType
8115 instances:
 - 8116 □ commodity
 - 8117 □ flowDirection
 - 8118 □ kind
 - 8119 □ uom
 - 8120 b) The following attributes SHOULD match in both servers' ReadingType instances or be
8121 accounted for (e.g., accumulationBehaviour attribute in Pricing server may be "Delta
8122 Data" whereas the Metering server may only provide "Cumulative"; these can be matched
8123 with the appropriate device logic) or have at least one be designated as "Not Specified":
 - 8124 □ accumulationBehaviour
 - 8125 □ phase
- 8126 5) If Rule #4 is not satisfied, clients SHOULD NOT display cost information to the user. Pricing
8127 clients SHOULD only display Pricing information. Pricing clients may indicate a mismatch
8128 between the Pricing and Metering server information and provide guidance to the user to contact
8129 his installer or service provider.

8130 17.2 PEV Implementation Guidelines (subject to work with SAE and ISO / IEC)

8131 Plug-In Electric Vehicles (PEVs) should implement Registration, DRLC, Pricing, and Messaging in order
8132 to alert the user of upcoming program events and to avoid charging during peak times. They may also
8133 implement FlowReservation in order to provide the ability to avoid high aggregated demand across
8134 multiple PEVs. They may also implement DER functionality in order to participate in programs that send
8135 advanced controls for grid conditioning and generation. Electric Vehicle Service Equipment (EVSE) may
8136 also use these functions, and may also use or implement metering to publish or obtain readings from
8137 another device in order to manage charging functions.