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

April 2013
This page is intentionally blank.
Notice of use and disclosure

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

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

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

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

ZigBee Alliance, Inc.                HomePlug Powerline Alliance, Inc.
2400 Camino Ramon, Suite 375       5200 SW Macadam Ave., Suite 470
San Ramon, CA 94583                 Portland, OR 97239
The empowerment of consumers to manage their usage and generation of energy is a critical feature of the Smart Grid and is a basis of innovation for new products and services in energy management. To enable this capability, information flow between devices such as meters, smart appliances, plug-in electric vehicles, energy management systems, and distributed energy resources (including renewable energy and storage elements) must occur in an open, standardized, secure, and interoperable fashion. The following specification is intended to fulfill those needs.

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

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

This document is also intended to enable communications that are link-layer agnostic and run over the Internet Protocol. Careful consideration was given to premises networks with various architectures, numbers of devices, and constraints while maintaining flexibility, extensibility, and security.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1</td>
<td>Certificate Fingerprint</td>
<td>49</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Short-form Device Identifier (SFDI)</td>
<td>49</td>
</tr>
<tr>
<td>8.3.3</td>
<td>Long-form Device Identifier (LFDI)</td>
<td>49</td>
</tr>
<tr>
<td>8.3.4</td>
<td>6-digit PIN code</td>
<td>49</td>
</tr>
<tr>
<td>8.3.5</td>
<td>Registration Code</td>
<td>50</td>
</tr>
<tr>
<td>8.4</td>
<td>Resource Access Authentication and Authorization context</td>
<td>50</td>
</tr>
<tr>
<td>8.5</td>
<td>Resource Access Authentication</td>
<td>51</td>
</tr>
<tr>
<td>8.6</td>
<td>Resource Access Authorization</td>
<td>52</td>
</tr>
<tr>
<td>8.7</td>
<td>Cipher Suites</td>
<td>52</td>
</tr>
<tr>
<td>8.8</td>
<td>Default Security Policy</td>
<td>52</td>
</tr>
<tr>
<td>8.9</td>
<td>Registration</td>
<td>54</td>
</tr>
<tr>
<td>8.9.1</td>
<td>EndDeviceList</td>
<td>55</td>
</tr>
<tr>
<td>8.10</td>
<td>Security LogEvents</td>
<td>56</td>
</tr>
<tr>
<td>8.11</td>
<td>Certificate Management</td>
<td>56</td>
</tr>
<tr>
<td>8.11.1</td>
<td>Introduction</td>
<td>56</td>
</tr>
<tr>
<td>8.11.2</td>
<td>Certificate Usage – Authentication vs. Authorization</td>
<td>58</td>
</tr>
<tr>
<td>8.11.3</td>
<td>Manufacturing PKI</td>
<td>59</td>
</tr>
<tr>
<td>8.11.4</td>
<td>General Certificate Format</td>
<td>60</td>
</tr>
<tr>
<td>8.11.5</td>
<td>General Restrictions and Conditions</td>
<td>61</td>
</tr>
<tr>
<td>8.11.6</td>
<td>Extensions</td>
<td>61</td>
</tr>
<tr>
<td>8.11.7</td>
<td>Additional ASN1 Definitions</td>
<td>62</td>
</tr>
<tr>
<td>8.11.8</td>
<td>Certificate Profiles</td>
<td>63</td>
</tr>
<tr>
<td>8.11.9</td>
<td>Device Requirements</td>
<td>67</td>
</tr>
<tr>
<td>8.11.10</td>
<td>Certificate Verification</td>
<td>68</td>
</tr>
<tr>
<td>8.11.11</td>
<td>Certificate Related Labeling Requirements</td>
<td>68</td>
</tr>
<tr>
<td>9</td>
<td>Discovery</td>
<td>69</td>
</tr>
<tr>
<td>9.1</td>
<td>Service Instance</td>
<td>69</td>
</tr>
<tr>
<td>9.2</td>
<td>Service Name</td>
<td>70</td>
</tr>
<tr>
<td>9.3</td>
<td>TXT Record</td>
<td>70</td>
</tr>
<tr>
<td>9.4</td>
<td>Subtype Queries</td>
<td>71</td>
</tr>
<tr>
<td>9.5</td>
<td>Discovery Procedure</td>
<td>73</td>
</tr>
<tr>
<td>10</td>
<td>Support Resources</td>
<td>74</td>
</tr>
<tr>
<td>10.1</td>
<td>Resource Section Outlines</td>
<td>74</td>
</tr>
<tr>
<td>10.1.1</td>
<td>Overview</td>
<td>74</td>
</tr>
<tr>
<td>10.1.2</td>
<td>List Ordering</td>
<td>74</td>
</tr>
<tr>
<td>10.1.3</td>
<td>Application Guidelines / Behavior</td>
<td>74</td>
</tr>
<tr>
<td>10.1.4</td>
<td>LogEvents</td>
<td>75</td>
</tr>
<tr>
<td>10.2</td>
<td>Device Capabilities Function Set</td>
<td>75</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Overview</td>
<td>75</td>
</tr>
<tr>
<td>10.2.2</td>
<td>List Ordering</td>
<td>75</td>
</tr>
<tr>
<td>10.2.3</td>
<td>Application Guidelines / Behavior</td>
<td>75</td>
</tr>
<tr>
<td>10.2.4</td>
<td>LogEvents</td>
<td>75</td>
</tr>
<tr>
<td>10.3</td>
<td>Self Device Resource</td>
<td>75</td>
</tr>
<tr>
<td>10.3.1</td>
<td>Overview</td>
<td>75</td>
</tr>
<tr>
<td>10.3.2</td>
<td>List Ordering</td>
<td>75</td>
</tr>
<tr>
<td>10.3.3</td>
<td>Application Guidelines / Behavior</td>
<td>75</td>
</tr>
<tr>
<td>10.3.4</td>
<td>LogEvents</td>
<td>76</td>
</tr>
<tr>
<td>10.4</td>
<td>End Device Resource</td>
<td>76</td>
</tr>
<tr>
<td>10.4.1</td>
<td>Overview</td>
<td>76</td>
</tr>
<tr>
<td>Page</td>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>204</td>
<td>10.4.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>205</td>
<td>10.4.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>206</td>
<td>10.4.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>207</td>
<td>10.5</td>
<td>Function Set Assignments</td>
</tr>
<tr>
<td>208</td>
<td>10.5.1</td>
<td>Overview</td>
</tr>
<tr>
<td>209</td>
<td>10.5.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>210</td>
<td>10.5.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>211</td>
<td>10.5.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>212</td>
<td>10.6</td>
<td>Subscription / Notification Mechanism</td>
</tr>
<tr>
<td>213</td>
<td>10.6.1</td>
<td>Overview</td>
</tr>
<tr>
<td>214</td>
<td>10.6.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>215</td>
<td>10.6.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>216</td>
<td>10.6.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>217</td>
<td>10.7</td>
<td>Response</td>
</tr>
<tr>
<td>218</td>
<td>10.7.1</td>
<td>Overview</td>
</tr>
<tr>
<td>219</td>
<td>10.7.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>220</td>
<td>10.7.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>221</td>
<td>10.7.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>222</td>
<td>11</td>
<td>Common Resources</td>
</tr>
<tr>
<td>223</td>
<td>11.1</td>
<td>Time Function Set</td>
</tr>
<tr>
<td>224</td>
<td>11.1.1</td>
<td>Overview</td>
</tr>
<tr>
<td>225</td>
<td>11.1.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>226</td>
<td>11.1.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>227</td>
<td>11.1.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>228</td>
<td>11.2</td>
<td>DeviceInformation Function Set</td>
</tr>
<tr>
<td>229</td>
<td>11.2.1</td>
<td>Overview</td>
</tr>
<tr>
<td>230</td>
<td>11.2.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>231</td>
<td>11.2.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>232</td>
<td>11.2.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>233</td>
<td>11.3</td>
<td>Power Status</td>
</tr>
<tr>
<td>234</td>
<td>11.3.1</td>
<td>Overview</td>
</tr>
<tr>
<td>235</td>
<td>11.3.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>236</td>
<td>11.3.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>237</td>
<td>11.3.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>238</td>
<td>11.4</td>
<td>Network Status</td>
</tr>
<tr>
<td>239</td>
<td>11.4.1</td>
<td>Overview</td>
</tr>
<tr>
<td>240</td>
<td>11.4.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>241</td>
<td>11.4.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>242</td>
<td>11.4.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>243</td>
<td>11.5</td>
<td>LogEvent List</td>
</tr>
<tr>
<td>244</td>
<td>11.5.1</td>
<td>Overview</td>
</tr>
<tr>
<td>245</td>
<td>11.5.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>246</td>
<td>11.5.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>247</td>
<td>11.6</td>
<td>Configuration Resource</td>
</tr>
<tr>
<td>248</td>
<td>11.6.1</td>
<td>Overview</td>
</tr>
<tr>
<td>249</td>
<td>11.6.2</td>
<td>List Ordering</td>
</tr>
<tr>
<td>250</td>
<td>11.6.3</td>
<td>Application Guidelines / Behavior</td>
</tr>
<tr>
<td>251</td>
<td>11.6.4</td>
<td>LogEvents</td>
</tr>
<tr>
<td>252</td>
<td>11.7</td>
<td>File Download Function Set</td>
</tr>
</tbody>
</table>
12 Smart Energy Resources ............................................................................................................... 96
12.1 Common Functionality ............................................................................................................... 96
  12.1.1 Overview ............................................................................................................................ 96
  12.1.2 Active List Elements ......................................................................................................... 96
  12.1.3 Event Rules and Guidelines .............................................................................................. 96
  12.1.4 Randomization .................................................................................................................. 99
  12.1.5 Multi-Server ..................................................................................................................... 101
12.2 Demand Response and Load Control ......................................................................................... 102
  12.2.1 Overview ........................................................................................................................ 102
  12.2.2 List Ordering ..................................................................................................................... 102
  12.2.3 Application Guidelines / Behavior .................................................................................. 103
  12.2.4 LogEvents ....................................................................................................................... 105
12.3 Metering Function Set .............................................................................................................. 105
  12.3.1 Overview ........................................................................................................................ 105
  12.3.2 List Ordering ..................................................................................................................... 105
  12.3.3 Application Guidelines / Behavior .................................................................................. 105
  12.3.4 LogEvents ....................................................................................................................... 109
12.4 Pricing Function Set .................................................................................................................. 111
  12.4.1 Overview ........................................................................................................................ 111
  12.4.2 List Ordering ..................................................................................................................... 111
  12.4.3 Application Guidelines / Behavior .................................................................................. 111
  12.4.4 LogEvents ....................................................................................................................... 115
12.5 Messaging Function Set .......................................................................................................... 115
  12.5.1 Overview ........................................................................................................................ 115
  12.5.2 List Ordering ..................................................................................................................... 115
  12.5.3 Application Guidelines / Behavior .................................................................................. 116
  12.5.4 LogEvents ....................................................................................................................... 116
12.6 Billing Function Set .................................................................................................................... 116
  12.6.1 Overview ........................................................................................................................ 116
  12.6.2 List Ordering ..................................................................................................................... 117
  12.6.3 Application Guidelines / Behavior .................................................................................. 117
  12.6.4 LogEvents ....................................................................................................................... 119
12.7 Prepayment Function Set ......................................................................................................... 119
  12.7.1 Overview ........................................................................................................................ 119
  12.7.2 List Ordering ..................................................................................................................... 119
  12.7.3 Application Guidelines / Behavior .................................................................................. 119
  12.7.4 LogEvents ....................................................................................................................... 121
12.8 Energy Flow Reservation Function Set ..................................................................................... 121
  12.8.1 Overview ........................................................................................................................ 121
  12.8.2 List Ordering ..................................................................................................................... 121
  12.8.3 Application Guidelines / Behavior .................................................................................. 121
  12.8.4 LogEvents ....................................................................................................................... 122
12.9 Distributed Energy Resources Function Set ............................................................................... 122
  12.9.1 Overview ........................................................................................................................ 122
  12.9.2 Terminology and Conventions ....................................................................................... 123
  12.9.3 List Ordering ..................................................................................................................... 124
  12.9.4 Application Guidelines / Behavior .................................................................................. 124
15.1.10 NetworkStatus Package ........................................................................................................ 189
15.1.11 LogEvents Package ............................................................................................................. 194
15.1.12 Configuration Package ........................................................................................................ 196
15.1.13 SoftwareDownload Package ............................................................................................... 198
15.1.14 DLRC Package .................................................................................................................... 201
15.1.15 Metering Package ................................................................................................................. 205
15.1.16 Pricing Package .................................................................................................................... 212
15.1.17 Messaging Package .............................................................................................................. 216
15.1.18 Billing Package .................................................................................................................... 218
15.1.19 Prepayment Package ........................................................................................................... 223
15.1.20 FlowReservation Package .................................................................................................. 228
15.1.21 DER Package ....................................................................................................................... 230
15.1.22 Links Package ...................................................................................................................... 246

16 Appendix C – Examples and Guidelines [INFORMATIVE] .................................................................. 252

16.1 Registration: Remote ................................................................................................................ 252
16.2 Registration: Local.................................................................................................................... 255
16.3 Discovery: Function Set Assignment ....................................................................................... 258
16.4 Discovery: Without Function Set Assignment ......................................................................... 260
16.5 Discovery: Undirected Without Function Set Assignment ..................................................... 262
16.6 Subscription / Notification ....................................................................................................... 263
16.7 Demand Response: General .................................................................................................... 266
16.8 Demand Response: Cancel ...................................................................................................... 270
16.9 Distributed Energy Resource: General .................................................................................... 272

16.10 Metering: Reading .................................................................................................................. 276
16.11 Metering: Interval.................................................................................................................... 281
16.12 Metering: Instantaneous .......................................................................................................... 287
16.13 Metering: Mirroring ............................................................................................................... 290
16.14 Pricing: Time of Use .............................................................................................................. 303
16.15 Billing: Billing Period ............................................................................................................. 309
16.16 Billing: Historical ................................................................................................................... 312
16.17 Billing: Projection ................................................................................................................... 315
16.18 File Loading ............................................................................................................................ 317
16.19 Flow Reservation: General ................................................................................................... 320
16.20 Flow Reservation: Cancel ..................................................................................................... 324
16.21 Event Randomization ............................................................................................................ 328

16.21.1 Simple Event ....................................................................................................................... 328
16.21.2 Event with Positive Randomized Duration ........................................................................ 329
16.21.3 Event with Positive Randomized Start ............................................................................... 329
16.21.4 Event with Negative Randomized Start ............................................................................ 330
16.21.5 Event with Positive Randomization and Finishing in one Hour ......................................... 331
16.21.6 Event with Negative Randomized Start and at Least One Hour Duration ...................... 331
16.21.7 Event with Randomized Start and Long Ramp Up ............................................................ 332
16.21.8 Event with Positive Randomized Start and Fixed End Time ............................................ 333

17 Appendix D – Guidelines [INFORMATIVE] .................................................................................. 334

17.1 Pricing Implementation Guidelines ........................................................................................ 334

17.1.1 Implementing Common Tariff Designs ............................................................................... 334
17.1.2 Flat Rate Design .................................................................................................................. 334
17.1.3 Time-of-Use Rate Design .................................................................................................... 335
17.1.4 Consumption-based Block Rate Design ................................................................................ 336
17.1.5 Combined Time-of-Use and Consumption-based Block Rate Design .......................................................... 336
17.1.6 Coordinated and Uncoordinated Pricing and Metering Servers ................................................................. 338
17.2 PEV Implementation Guidelines (subject to work with SAE and ISO / IEC) ..................................................... 339
List of Tables

404 Table 7-1 HTTP Headers. ................................................................................. 36
405 Table 8-1 Local Registration Attributes.............................................................. 43
406 Table 8-2 Local Registration Descriptor Entry.................................................... 43
407 Table 8-3 ACL Attributes.................................................................................. 44
408 Table 8-4 AccessDescriptor Entry...................................................................... 44
409 Table 8-5 SpecificIDDescriptor Entry................................................................. 45
410 Table 8-6 Example ACL for EndDeviceList....................................................... 47
411 Table 8-7 Example ACL entry for EndDevice at point of registration. .............. 47
412 Table 8-8 Example local registration entry for registering device..................... 48
413 Table 8-9 Example ACL entry for EndDevice at point of client access. .......... 48
414 Table 8-10 Example SpecificIDDescriptor for EndDevice at point of client access .............................................................................................................. 48
415 Table 8-11 Attribute values for Default Security Policy ...................................... 53
416 Table 8-12 Security LogEvents........................................................... 56
417 Table 8-13 TLS Authentication Matrix .................................................. 58
418 Table 8-14 SEP 2 Authorization Matrix...................................................... 58
419 Table 9-1 TXT record parameters. .................................................................. 70
420 Table 9-2 Service subtype strings and their corresponding SEP 2 function sets. .............................................................................................................. 72
421 Table 10-1 Function Set List Ordering. ................................................... 74
422 Table 10-2 Example LogEvents. .................................................................... 75
423 Table 10-3 Self Device LogEvents. ................................................................. 76
424 Table 10-4 End Device List Ordering.............................................................. 76
425 Table 10-5 End Device LogEvents. ................................................................. 77
426 Table 10-6 Function Set Assignments List Ordering...................................... 78
427 Table 10-7 Subscription / Notification List Ordering. .................................... 79
428 Table 10-8 Subscription / Notification LogEvents .......................................... 81
429 Table 10-9 Response List Ordering ............................................................... 81
430 Table 10-10 Response Types by Function Set .............................................. 82
431 Table 10-11 Response LogEvents................................................................. 85
432 Table 11-1 Time LogEvents. .......................................................................... 87
433 Table 11-2 Power Status LogEvents............................................................... 88
434 Table 11-3 Network Status List Ordering ..................................................... 88
435 Table 11-4 Network Status LogEvents ......................................................... 89
436 Table 11-5 LogEvent List Ordering ................................................................ 89
437 Table 11-6 General LogEvents .................................................................... 90
438 Table 11-7 Configuration LogEvents .............................................................. 91
439 Table 11-8 File Download List Ordering ......................................................... 91
440 Table 11-9 File Download LogEvents ............................................................. 94
441 Table 12-1 Demand Response and Load Control List Ordering............... 102
442 Table 12-2 Metering List Ordering ................................................................ 105
443 Table 12-3 Reading Types .............................................................................. 108
444 Table 12-4 Metering LogEvents ................................................................. 109
445 Table 12-5 Pricing List Ordering .................................................................... 111
446 Table 12-6 Pricing LogEvents ....................................................................... 115
447 Table 12-7 Messaging List Ordering ................................................................. 115
448 Table 12-8 Billing List Ordering ....................................................................... 117
449 Table 12-9 Prepayment List Ordering ............................................................. 119
450 Table 12-10 Prepayment LogEvents ............................................................... 121
451 Table 12-11 FlowReservation List Ordering .................................................... 121
Table 12-12 Flow Reservation LogEvents.............................................................................................................. 122
Table 12-13 Distributed Energy Resources List Ordering.................................................................................................. 124
Table 12-14 Modes and Attributes for Generator Type DERs.......................................................................................... 128
Table 12-15 Modes and Attributes for Storage Type DERs............................................................................................ 129
Table 12-16 Metering Mirror List Ordering.................................................................................................................. 129
Table 12-17 Metering Mirror LogEvents...................................................................................................................... 131
Table 16-1 POX Example: Registration Remote........................................................................................................... 252
Table 16-2 EXI Example: Registration Remote............................................................................................................... 254
Table 16-3 POX Example: Registration Local................................................................................................................ 256
Table 16-4 POX Example: Discovery Function Set Assignment.......................................................................................... 258
Table 16-5 POX Example: Discovery Without Function Set Assignment........................................................................ 260
Table 16-6 POX Example: Discovery Undirected Without Function Set Assignment....................................................... 262
Table 16-7 POX Example: Subscription / Notification.................................................................................................. 264
Table 16-8 EXI Example: Subscription / Notification.................................................................................................... 265
Table 16-9 POX Example: Demand Response – General................................................................................................. 266
Table 16-10 POX Example: Demand Response - Cancel................................................................................................. 270
Table 16-11 POX Example: Distributed Energy Resource - General.................................................................................. 273
Table 16-12 POX Example: Meter Reading.................................................................................................................... 277
Table 16-13 POX Example: Metering Interval................................................................................................................ 282
Table 16-14 POX Example: Metering Instantaneous..................................................................................................... 287
Table 16-15 POX Example: Meter Mirroring.................................................................................................................. 291
Table 16-16 POX Example: Pricing TOU......................................................................................................................... 304
Table 16-17 POX Example: Billing Period...................................................................................................................... 309
Table 16-18 POX Example: Billing Historical................................................................................................................ 312
Table 16-19 POX Example: Billing Projection................................................................................................................ 315
Table 16-20 File Load: Flow Description....................................................................................................................... 318
Table 16-21 POX Example: Flow Reservation - General.................................................................................................. 321
Table 16-22 POX Example: Flow Reservation - Cancel................................................................................................ 325
Table 16-23 Ramp Value.................................................................................................................................................. 328
Table 16-24 Event Ramp Time........................................................................................................................................... 329
Table 16-25 Event Start Time............................................................................................................................................ 330
Table 16-26 Event Start Time............................................................................................................................................ 330
Table 16-27 Event Start Time............................................................................................................................................ 331
Table 16-28 Event Start Time............................................................................................................................................ 331
Table 16-29 Event Start Time............................................................................................................................................ 332
Table 17-1 TOU Rate......................................................................................................................................................... 335
Table 17-2 Consumption-based Rates............................................................................................................................ 336
Table 17-3 Tariff Structure................................................................................................................................................ 336

Copyright © 2013, ZigBee Alliance, Inc. and HomePlug Powerline Alliance, Inc. All rights reserved.
List of Figures

Figure 8-1: Example Device Authentication Procedure Using HTTPS Port 443 ........................................... 51
Figure 8-2: Device authentication with registration procedure examples using HTTPS ............................... 55
Figure 8-3: SEP 2 including ZigBee IP deployment...................................................................................... 57
Figure 12-1: Active and Reactive Power Flow Directions as Measured at the DER .................................... 124
Figure 12-2: Example Volt-Var Curve and Hysteresis ................................................................................. 126
Figure 12-3: Example Low and High - Voltage Ride Through Curves .......................................................... 126
Figure 13-1: Allowed and Disallowed Extension ......................................................................................... 132
Figure 15-1: Version Information .................................................................................................................. 155
Figure 15-2: DeviceCapability ...................................................................................................................... 155
Figure 15-3: Identification ............................................................................................................................ 156
Figure 15-4: Events ....................................................................................................................................... 159
Figure 15-5: Programs ................................................................................................................................... 159
Figure 15-6: Error ......................................................................................................................................... 160
Figure 15-7: Types ......................................................................................................................................... 162
Figure 15-8: Primitive Types ......................................................................................................................... 173
Figure 15-9: SelfDevice ................................................................................................................................. 175
Figure 15-10: EndDevice ............................................................................................................................... 176
Figure 15-11: FunctionSetAssignments ...................................................................................................... 178
Figure 15-12: Pub-Sub .................................................................................................................................. 179
Figure 15-13: Response ................................................................................................................................. 181
Figure 15-14: Time ........................................................................................................................................ 183
Figure 15-15: DeviceInformation ................................................................................................................. 184
Figure 15-16: PowerStatus ........................................................................................................................... 187
Figure 15-17: NetworkStatus ....................................................................................................................... 189
Figure 15-18: LogEvents ............................................................................................................................... 194
Figure 15-19: Configuration ........................................................................................................................ 196
Figure 15-20: Files ........................................................................................................................................ 198
Figure 15-21: DRLC Event ............................................................................................................................ 201
Figure 15-22: Load Shed Availability ......................................................................................................... 202
Figure 15-23: Metering Data ......................................................................................................................... 205
Figure 15-24: Metering Data Types ............................................................................................................ 206
Figure 15-25: Metering Mirror ................................................................................................................... 209
Figure 15-26: Metering Mirror Inheritance ................................................................................................. 210
Figure 15-27: Pricing ..................................................................................................................................... 212
Figure 15-28: TextMessage .......................................................................................................................... 216
Figure 15-29: Billing ...................................................................................................................................... 218
Figure 15-30: Billing Readings...................................................................................................................... 219
Figure 15-31: Prepayment ............................................................................................................................ 223
Figure 15-32: FlowReservation .................................................................................................................. 228
Figure 15-33: DER Info ................................................................................................................................. 230
Figure 15-34: DER Info Types ..................................................................................................................... 231
Figure 15-35: DER Program ........................................................................................................................ 231
Figure 15-36: DER Curves ............................................................................................................................ 232
Figure 15-37: DER Control .......................................................................................................................... 233
Figure 15-38: DER Control Types ............................................................................................................. 234
Figure 16-1: Remote Registration ................................................................................................................ 252
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>539</td>
<td>Figure 16-2: Registration Local</td>
</tr>
<tr>
<td>540</td>
<td>Figure 16-3: Discovery: Function Set Assignment</td>
</tr>
<tr>
<td>541</td>
<td>Figure 16-4: Discovery Without Function Set Assignment</td>
</tr>
<tr>
<td>542</td>
<td>Figure 16-5: Discovery Undirected Without Function Set Assignment</td>
</tr>
<tr>
<td>543</td>
<td>Figure 16-6: Subscription / Notification</td>
</tr>
<tr>
<td>544</td>
<td>Figure 16-7: Demand Response General</td>
</tr>
<tr>
<td>545</td>
<td>Figure 16-8: Demand Response - Cancel</td>
</tr>
<tr>
<td>546</td>
<td>Figure 16-9: Distributed Energy Resource: General</td>
</tr>
<tr>
<td>547</td>
<td>Figure 16-10: Meter Reading</td>
</tr>
<tr>
<td>548</td>
<td>Figure 16-11: Metering Interval</td>
</tr>
<tr>
<td>549</td>
<td>Figure 16-12: Metering Instantaneous</td>
</tr>
<tr>
<td>550</td>
<td>Figure 16-13: Meter Mirroring</td>
</tr>
<tr>
<td>551</td>
<td>Figure 16-14: Pricing Time of Use</td>
</tr>
<tr>
<td>552</td>
<td>Figure 16-15: Billing Period</td>
</tr>
<tr>
<td>553</td>
<td>Figure 16-16: Billing Historical</td>
</tr>
<tr>
<td>554</td>
<td>Figure 16-17: Billing Projection</td>
</tr>
<tr>
<td>555</td>
<td>Figure 16-18: File Load - FlowDiagram</td>
</tr>
<tr>
<td>556</td>
<td>Figure 16-19: Flow Reservation: General</td>
</tr>
<tr>
<td>557</td>
<td>Figure 16-20: Flow Reservation: Cancel</td>
</tr>
<tr>
<td>558</td>
<td>Figure 16-21: Simple Event - No Ramp Up - No Ramp Down</td>
</tr>
<tr>
<td>559</td>
<td>Figure 16-22: Event with Positive Randomization Duration</td>
</tr>
<tr>
<td>560</td>
<td>Figure 16-23: Event with Positive Randomization Start</td>
</tr>
<tr>
<td>561</td>
<td>Figure 16-24: Event with Negative Randomized Start</td>
</tr>
<tr>
<td>562</td>
<td>Figure 16-25: Positive Randomization in an Hour</td>
</tr>
<tr>
<td>563</td>
<td>Figure 16-26: Event with Negative Start in One Hour</td>
</tr>
<tr>
<td>564</td>
<td>Figure 16-27: Randomized Start and Long Ramp Up</td>
</tr>
<tr>
<td>565</td>
<td>Figure 16-28: Randomized Start and Fixed End Time</td>
</tr>
<tr>
<td>566</td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Purpose

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

1.2 Scope

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

1.3 Context Overview

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

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

1.4 Document Organization

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

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

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

1.5 Requirement Language

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

1.6 Typography Conventions Used

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

1.7 Design Principles

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

- 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.
2 Acknowledgements

On behalf of the ZigBee Alliance, the HomePlug Powerline Alliance, and all of our liaison partners, we would like to extend our thanks to all who participated in the development of this specification. Representatives from many organizations have participated in weekly calls, face-to-face meetings, email exchanges, and specialized editing sessions. Without the broad support from both member and external organizations that care about the development of smart energy solutions, this work would not have been possible. Select individuals stand out for their contributions to this document.

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

Robby Simpson: Chair
Tim Gillman: Vice-chair
Jeff Shudark: Secretary
Don Sturek: Program Manager
Tobin Richardson: ZigBee Alliance Chairman and CEO
Steven Van Ausdall: Model, XSD, and WADL Editor
Al Petrick: Application Specification Editor

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

Casey Anderson
Skip Ashton
Dan Bailey
Gary Birk
Donald Bowen
Doug Burman
Paul Carreon
Stephen Chasko
Michael Cowan
Robert Cragie
Wayne Dennison
Yusuke Doi
Jeff Drake
Paul Duffy

Ed Eckert
Gene Falendysz
Matthew Gillmore
Tom Herbst
Lance Hester
Ken Holbrook
Shawn Hu
Rajagopal Iyengar
Jeffrey King
David Kravitz
Chris Leclercq
Kerry Lynn
Zahra Makoui
Tony Mauro

Ricardo Montano
Tim Moses
Leslie Mulder
Don Olsen
Ivan O'Neill
Barry Peirce
Joseph Reddy
Emil Romascanu
David Smith
Michael StJohns
Michael Stuber
Mark Ward

Page 19
### Document Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2.0</td>
<td>Smart Energy Profile 2</td>
</tr>
</tbody>
</table>
4 References

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

4.1 Smart Energy 2.0 Documents


4.2 IEC Documents

[61850] ....................... Communication networks and systems in substations - ALL PARTS
(http://webstore.iec.ch).


4.3 IETF Documents


[RFC 2119] ....................... Key words for use in RFCs to Indicate Requirement Levels


[RFC 2717] ....................... Registration Procedures for URL Scheme Names


[RFC 3635] Definitions of Management Objects for the Internet-Like Interface Types

[RFC 4108] Using Cryptographic Message Syntax (CMS) to Protect Firmware,


[RFC 4293] Management Information Base for the Internet Protocol (IP)


[RFC 5280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation
List (CRL) Profile, IETF, D Cooper et al, May 2008,

[RFC 5480] Elliptic Curve Cryptography Subject Public Key Information, IETF, S. Turner


[RFC 6554] An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-


[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).

[I-D XMDNS] Extended Multicast DNS (https://docs.zigbee.org/zigbee-docs/dcn/12/docs-12-
0461-00-SEP-2-xmdns.pdf).

4.4 Other References


Number 1023059 (http://my.epri.com/portal/server.pt?space=CommunityPage&cached=true&pare
ntname=ObjMgr&parentid=2&control=SetCommunity&CommunityID=221&Pa
geIDQueryComId=0).

[EXI] Efficient XML Interchange (EXI) Format 1.0 (http://www.w3.org/TR/2008/WD-
exi-20080919/).
[EXIBP] Efficient XML Interchange (EXI) best practices (http://www.w3.org/TR/exi-best-practices/).


[Linux] Linux (https://www.linux.com/).


[PEN] Internet Authority Assignment numbers (IANA) PEN (http://pen.iana.org/pen/PenApplication.page).


[SunSpec] SunSpec Alliance Inverter Control Model, work in progress. (http://www.sunspec.org/specifications/).

[WADL] Web Application Description Language (http://www.w3.org/Submission/wadl/).

5 Definitions, Acronyms and Abbreviations

5.1 Acronyms and Abbreviations

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

ACL .......... Access Control List
CA............ Certificate Authority
CRL.......... Certificate Revocation List
CSEP........ Consortium for SEP 2 Interoperability
DER......... Distributed Energy Resource
DNS......... Domain Name System
DNS-SD..... DNS-based Service Discovery
DRLC........ Demand Response and Load Control
ECDSA...... Elliptic Curve Digital Signature Algorithm
EMS......... Energy Management System
ESI......... Energy Services Interface
EVSE........ Electric Vehicle Supply Equipment
EXI......... Efficient XML Interchange
HTTP........ Hypertext Transfer Protocol
IETF......... Internet Engineering Task Force
IP.......... Internet Protocol
LFDI......... Long Form Device Identifier
MCA......... Manufacturer's CA
MICA........ Manufacturing Issuing CA
OCSP......... Online Certificate Status Protocol
OID......... Object Identifier
PKI......... Public Key Infrastructure
REST........ Representational State Transfer
SERCA...... Smart Energy Root CA
SFDI......... Short Form Device Identifier
TCP......... Transmission Control Protocol
UOM......... Units of Measure
URI......... Uniform Resource Identifier
XML......... Extensible Markup Language
WADL ...... Web Application Description Language

5.2 Definitions

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

ACCESS CONTROL LIST

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

CERTIFICATE AUTHORITY

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

CERTIFICATE CHAIN

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

CLIENT

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

DEVICE CERTIFICATE

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

ENERGY SERVICES INTERFACE (ESI)

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

FINGERPRINT

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

FUNCTION SET

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

FUNCTION SET ASSIGNMENTS

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

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

**INTERMEDIATE CA**

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

**ISSUING CA**

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

**MANUFACTURER'S CA (MCA)**

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

**MANUFACTURING ISSUING CA (MICA)**

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

**MANUFACTURING PKI**

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

**NODE**

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

**OID**

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

**PEM FORMAT CERTIFICATE**

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

**PKI**

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

**REGISTERED**

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

**RESOURCE**

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

**RESOURCE DISCOVERY**

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

A CA whose certificate or public key is a trust anchor for any other certificates in a chain of trust.

ROOT CERTIFICATE

The Root CA's self-signed certificate. Generally also a Trust Anchor.

SELF-SIGNED CERTIFICATE

A certificate whose issuer and subject are identical, and whose public key verifies its signature.

SERVER

The device or host that holds a resource and exposes representations of that resource.

SMART ENERGY ROOT CA (SERCA)

The top-level CA for the SEP 2 Manufacturing PKI.

TRUST ANCHOR

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.

TRUSTED ROOT STORE

An integrity-protected location for storing root certificates.
6 Design Pattern

6.1 Protocol Flexibility

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

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

6.2 General Rules / Best Practices

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

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

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

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

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

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

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

Devices do not assume the use of the URIs and their structures given throughout this specification. All resources are self-describing as it is acknowledged that URIs, schemas, and resources might change in the future. All resources SHALL contain links to their subordinate resources to support flexibility in URIs and future extensibility. Thus, to allow for extensibility and granularity, all objects are described in schemas and referenced via URIs. The URIs presented throughout this specification are recommendations. Thus, clients do not assume that URIs for resources are fixed on all servers or even on a given server (over time), but rather retrieve the appropriate URIs through resource discovery and links within resources. For network efficiency, devices MAY assume URIs are fixed on a particular server over time. If a URI returns an unexpected result, the client SHOULD execute resource discovery to determine the new URI value.
Version information should not be presented in the URI unless that version information is inherent to the name of that resource. If necessary and for reasons of extensibility, version information is provided within the associated resources and / or schemas.

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

6.3 WADL

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

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

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

6.4 Schema

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

6.5 Uniform Resource Identifiers

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

The following conventions are used for URI naming:

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

4) URIs SHALL NOT be greater than 255-bytes in length. In practice, URIs SHOULD be much smaller than 80-bytes.

6.6 **List Resources**

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

The following attributes are defined for list resources:

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

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

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

1) Ordinal access to the first, second, nth, etc. item in the list is supported via query string parameters included with a GET to the list resource URI.

2) Random access to specific list items is supported via a GET directly to the URI of the list item.

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

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

6.6.1 **Query String Parameters**

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

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

\[
\text{URI}\?s=\{x\}&a=\{y\}&l=\{z\}
\]

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

The query string parameters are defined as:

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


The MyTypeList resource, depicted below:

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

```
<MyTypeList href="http://host1/the/list" all="?" results="7">
  <MyType href="http://host1/instance/of/type/red">
    <TimeStamp>100</TimeStamp>
  </MyType>
  <MyType href="http://host2/instance/of/type/green">
    <TimeStamp>200</TimeStamp>
  </MyType>
  <MyType href="http://host3/instance/of/type/blue">
    <TimeStamp>300</TimeStamp>
  </MyType>
  <MyType href="http://host4/instance/of/type/yellow">
    <TimeStamp>400</TimeStamp>
  </MyType>
  <MyType href="http://host5/instance/of/type/black">
    <TimeStamp>500</TimeStamp>
  </MyType>
</MyTypeList>
```

- 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 SHALL 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:

```
<MyTypeList href="http://host1/the/list" all="?" results="7">
  <MyType href="http://host1/instance/of/type/red">
    <TimeStamp>100</TimeStamp>
  </MyType>
  <MyType href="http://host2/instance/of/type/green">
    <TimeStamp>200</TimeStamp>
  </MyType>
  <MyType href="http://host3/instance/of/type/blue">
    <TimeStamp>300</TimeStamp>
  </MyType>
  <MyType href="http://host4/instance/of/type/yellow">
    <TimeStamp>400</TimeStamp>
  </MyType>
  <MyType href="http://host5/instance/of/type/black">
    <TimeStamp>500</TimeStamp>
  </MyType>
</MyTypeList>
```
This list is sorted in ascending order per the <timeStamp/> element of MyType.

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

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

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

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

A GET to http://host1/the/list?s=12&l=2 will return a list representation containing 0 items:
A GET to http://host6/instance/of/type/white will return:

```xml
<MyType href="http://host6/instance/of/type/white">
  <timeStamp>600</timeStamp>
</MyType>
```

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

```xml
<MyTypeList href="http://host1/the/list" all="7" results="3">
  <MyType href="http://host5/instance/of/type/black">
    <timeStamp>500</timeStamp>
  </MyType>
  <MyType href="http://host6/instance/of/type/white">
    <timeStamp>600</timeStamp>
  </MyType>
  <MyType href="http://host7/instance/of/type/orange">
    <timeStamp>700</timeStamp>
  </MyType>
</MyTypeList>
```

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

```xml
<MyTypeList href="http://host1/the/list" all="7" results="2">
  <MyType href="http://host5/instance/of/type/black">
    <timeStamp>500</timeStamp>
  </MyType>
  <MyType href="http://host6/instance/of/type/white">
    <timeStamp>600</timeStamp>
  </MyType>
</MyTypeList>
```

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

```xml
<MyTypeList href="http://host1/the/list" all="7" results="1">
  <MyType href="http://host7/instance/of/type/orange">
    <timeStamp>700</timeStamp>
  </MyType>
</MyTypeList>
```

### 6.7 Resource Design Rules

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

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

- Non-list resources SHALL NOT support the defined query string parameters. Query string 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.
SEP 2 provides a set of Mandatory, Optional, and Discouraged recommendations for each HTTP header. SEP 2 implementations SHOULD employ only Mandatory HTTP headers, minimize the use of Optional HTTP headers, and avoid the use of Discouraged headers.

### 7.4.1 HTTP Header Field Recommended Usage

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

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

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

<table>
<thead>
<tr>
<th>Header</th>
<th>Used in Message Type</th>
<th>RFC Required / Optional</th>
<th>SEP 2 Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Request</td>
<td>Optional</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Accept-Charset</td>
<td>Request</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>Request</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>Request</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Accept-Ranges</td>
<td>Request Response</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Age</td>
<td>Response</td>
<td>Optional (Required for a cache)</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Allow</td>
<td>Response</td>
<td>Required</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Authorization</td>
<td>Request Response</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Cache-Control</td>
<td>Request Response</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Connection</td>
<td>Request</td>
<td>Optional (Required in some situations (e.g., HTTP/1.1 applications that do not support persistent connections))</td>
<td>Optional (Mandatory in some situations)</td>
</tr>
<tr>
<td>Content-Encoding</td>
<td>Response</td>
<td>Optional (Required when an encoding is applied)</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Content-Language</td>
<td>Response</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Content-Length</td>
<td>Request Response</td>
<td>Optional (Required in many situations, see section 4.4 of [RFC 2616])</td>
<td>Optional (Mandatory in many situations, see section 4.4 of [RFC 2616])</td>
</tr>
<tr>
<td>Content-Location</td>
<td>Response</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Content-MD5</td>
<td>Response</td>
<td>Optional</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Header</td>
<td>Used in Message Type</td>
<td>RFC Required / Optional</td>
<td>SEP 2 Use</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------</td>
<td>-------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Content-Range</td>
<td>Request</td>
<td>Optional</td>
<td>Optional (see Section 11.7.1.4)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Request, Response</td>
<td>Required, Optional</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Cookies</td>
<td></td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Date</td>
<td>Request, Response</td>
<td>Mandatory, Mandatory</td>
<td>Mandatory</td>
</tr>
<tr>
<td>ETag</td>
<td>Response</td>
<td>Optional, Discouraged</td>
<td>Optional (see Section 11.7.1.4)</td>
</tr>
<tr>
<td>Expect</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Expires</td>
<td>Request, Response</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>From</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Host</td>
<td>Request</td>
<td>Required, Mandatory</td>
<td>Mandatory</td>
</tr>
<tr>
<td>If-Match</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>If-Modified-Since</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>If-None-Match</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>If-Range</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>If-Unmodified-Since</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Last-Modified</td>
<td>Response</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Location</td>
<td>Response</td>
<td>Optional, Discouraged</td>
<td>Mandatory in many situations (e.g., POST responses)</td>
</tr>
<tr>
<td>Max-Forwards</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Pragma</td>
<td>Request, Response</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td>Response</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Proxy-Authorization</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Range</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Optional (see Section 11.7.1.4)</td>
</tr>
<tr>
<td>Referrer</td>
<td>Request</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Retry-After</td>
<td>Response</td>
<td>Optional, Discouraged</td>
<td>Optional (see Section 11.7.1.4)</td>
</tr>
<tr>
<td>Server</td>
<td>Response</td>
<td>Optional, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>TE</td>
<td>Request</td>
<td>Discouraged, Discouraged</td>
<td>Discouraged</td>
</tr>
<tr>
<td>Trailer</td>
<td>Response</td>
<td>Discouraged, Discouraged</td>
<td>Discouraged</td>
</tr>
</tbody>
</table>
### 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

### 7.5 HTTP Response Codes

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

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

#### 7.5.1 Common Responses

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

**7.5.1.1 1xx (Informational)**

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

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

**7.5.1.2 200 ("OK")**

This response code is sent to indicate a successful transaction.

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

**7.5.1.3 201 ("Created")**

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

The Location header SHALL be used in conjunction with this response code to indicate the URI of the newly created resource. The inclusion of a representation of the newly created resource in the entity-body of the response is discouraged, to conserve bandwidth.
[RFC 2616] states, "If a new resource is created, the origin server MUST inform the user agent via the 201 (Created) response."

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

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

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

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

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

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

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

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

7.5.1.7  **302 ("Redirect")**

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

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

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

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

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

7.5.1.9  **401 ("Unauthorized")**

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

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

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

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

This response code is used to indicate that no resource can be found at the specified URI.
This response code MAY also be used in lieu of a 401 response code.

7.5.1.11  405 ("Method Not Allowed")

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

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

7.5.1.12  406 ("Not Acceptable")

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

7.5.1.13  413 ("Request Entity Too Large")

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

7.5.1.14  416 ("Requested Range Not Satisfiable")

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

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

7.5.1.15  417 ("Expectation Failed")

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

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

7.5.1.16  500 ("Internal Server Error")

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

7.5.1.17  501 ("Not Implemented")

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

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

7.5.1.18  503 ("Service Unavailable")

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

7.5.2  Minimal Understanding

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

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

7.6.1 XML Encoding

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

7.6.2 EXI Encoding

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

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

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

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

7.7 Content Negotiation

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

7.7.1 Schema Version Negotiation

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

The Extensibility Level defines the base schema and its capability for arbitrary extension. The Extensibility Level is one of "-S0" or "+S0". The S0 indicates the base schema version: SEP 2. "-S0"
indicates the node does not accept arbitrary tags that are not defined in the base schema, and "+S0"
indicates it accepts arbitrary tags. A node with "-S0" will likely fail on an EXI document using arbitrary
types, elements, and attributes that are not defined in the schema used for encoding. Devices SHALL
NOT send messages to nodes that declare "-S0" using arbitrary types, elements, and attributes.
The grammar used for EXI SHALL be generated as a non-strict grammar only, as having both strict and
non-strict grammars would put a large burden on storage requirements for certain devices. The use of a
non-strict grammar allows for extensions without schema modification. An invalid (i.e., not defined in
the schema) part of an EXI document is allowed in a non-strict grammar and can carry arbitrary tags,
attributes, and text encoded using the built-in grammar.
Due to strict memory constraints, some nodes may not be able to parse invalid parts of an EXI document
encoded using the built-in grammar. To avoid such errors, a node may declare its inability to receive
arbitrary extensions using the ":-" (minus) prefix in the Extensibility Level. Alternatively, nodes that
declare the "+" (plus) prefix in the Extensibility Level will be able to parse extended parts of an EXI
document.
Note that the Extensibility Level does not indicate whether the node can process the data, but only
whether it can parse the data.
The format of the Extensibility Level is "([+-]Sn)" where n is a character to describe base schema version
(currently "0"). As extensions of SEP 2.x schemas are intended to be backward compatible, a node that
declares schemaId "S[i]" is intended to be compatible with all versions between "S0" and "S[i]".
For example:
Accept: application/sep-exi; level=-S0
indicates that the client wishes to receive content encoded using EXI where the base schema of the client
is SEP 2 and that the client does not accept arbitrary tags not defined in the schema.
A client SHOULD use the Extensibility Level discovered during resource discovery to determine if a
server accepts non-strict parts of an EXI document prior to initiating PUT / POST operations where the
content contains extended attributes / elements. A server SHOULD use the Extensibility Level specified
in the Accept header to determine if a client accepts non-strict parts of an EXI document prior to
responding to GET operations where the content contains extended attributes / elements.
8 Security

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

8.1 Introduction

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

8.2 Security Attributes

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

8.2.1 Local Registration Attributes

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

Table 8-1 Local Registration Attributes.

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

Table 8-2 Local Registration Descriptor Entry.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN</td>
<td>Integer</td>
<td>0 – 999999</td>
<td>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</td>
<td>-</td>
</tr>
<tr>
<td>SFDI</td>
<td>SFDI</td>
<td>-</td>
<td>SFDI of registering device. The SFDI is also reflected in the EndDevice resource</td>
<td>-</td>
</tr>
<tr>
<td>DeviceType</td>
<td>Integer</td>
<td>0 – 3</td>
<td>Minimum required device type</td>
<td>0</td>
</tr>
<tr>
<td>HardwareModuleName</td>
<td>String</td>
<td>-</td>
<td>Optional additional hardware module information</td>
<td>-</td>
</tr>
</tbody>
</table>
8.2.2 **Access Control List (ACL) Attributes**

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

**Table 8-3 ACL Attributes.**

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

**Table 8-4 AccessDescriptor Entry.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Bitmap</td>
<td>0x0 – 0xf</td>
<td>Bitmap of which methods are supported:</td>
<td>0x0</td>
</tr>
<tr>
<td>AuthType</td>
<td>Integer</td>
<td>0x0 – 0xf</td>
<td>Bitmap of which authentication types are allowed:</td>
<td>0x0</td>
</tr>
<tr>
<td>DeviceType</td>
<td>Integer</td>
<td>0 – 3</td>
<td>Device type:</td>
<td>0</td>
</tr>
</tbody>
</table>

Remaining bits are reserved for authentication types not defined by this specification but by an additional security policy.
Table 8-5 SpecificIDDescriptor Entry.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Access Descriptor</td>
<td>-</td>
<td>Access levels required for client</td>
<td>-</td>
</tr>
<tr>
<td>IPAddr</td>
<td>IPAddress</td>
<td>-</td>
<td>IP Address of client</td>
<td>-</td>
</tr>
<tr>
<td>Port</td>
<td>integer</td>
<td>0x0000 – 0xffff</td>
<td>Port of client</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

Subordinate resources created dynamically will initially inherit their ACL from their parent resource. ACLs may be statically fixed with a default operation for some resources and may be dynamic and extensible for others. If a resource does not have an ACL, access is granted to the resource unconditionally.

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

8.2.2.1 aclDefaultAccess

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

8.2.2.2 aclSpecificIDList

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

8.2.2.3 Access Authorization

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

In the following, 'corresponding ACL entry' means:

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

8.2.2.3.1 Method Attribute

The Method attribute is used to control which HTTP request method is allowed for client access.
The HTTP method of an incoming HTTP request is checked and Method authorization will be TRUE if the method is allowed in the corresponding ACL entry.

8.2.2.3.2 **AuthType Attribute**

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

- 0x1: No authentication
- 0x2: User authentication
- 0x4: Self-signed Certificate
- 0x8: Device Certificate

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

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

```
if (authentication type & AuthType) != 0:
    AuthType authorization = TRUE
else
    AuthType authorization = FALSE
```

8.2.2.3.3 **DeviceType Attribute**

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

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

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

8.2.2.4 **Authorization Logic**

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

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

HTTP/1.1 400 Bad Request
If AuthType or DeviceType authorization is not granted, the server SHOULD immediately respond:

HTTP/1.1 404 Not Found

The server MAY respond:

HTTP/1.1 401 Unauthorized

Otherwise, processing will continue.

**ACL Examples**

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

### 8.2.2.5.1 EndDeviceList

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

Table 8-6 Example ACL for EndDeviceList.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>aclDefaultAccess</td>
<td>Method: 0x01 (GET only)</td>
</tr>
<tr>
<td></td>
<td>AuthType: 0x1 (no authentication)</td>
</tr>
<tr>
<td></td>
<td>Device Type: 0 (unknown)</td>
</tr>
<tr>
<td>aclSpecificID</td>
<td>Empty</td>
</tr>
<tr>
<td>aclSpecificIDEntries</td>
<td>0</td>
</tr>
</tbody>
</table>

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

### 8.2.2.5.2 EndDevice

In this example, an EndDevice resource for a client does not exist prior to registration. The earliest point it can be created is at the point of registration and the ACL would be set as follows:

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>aclDefaultAccess</td>
<td>Method: 0x00 (no default access)</td>
</tr>
<tr>
<td></td>
<td>AuthType: 0x1 (no authentication)</td>
</tr>
<tr>
<td></td>
<td>Device Type: 0 (unknown)</td>
</tr>
<tr>
<td>aclSpecificID</td>
<td>Empty</td>
</tr>
<tr>
<td>aclSpecificIDEntries</td>
<td>0</td>
</tr>
</tbody>
</table>

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

There would also be an entry placed in the local registration list corresponding to the client:
### Table 8-8 Example local registration entry for registering device.

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Access Descriptor</td>
<td>-</td>
<td>Access levels required for client: Method: 0xF (GET, PUT, POST, DELETE)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AuthType: 0x8 (Device Certificate) DeviceType: (as appropriate for device)</td>
<td></td>
</tr>
<tr>
<td>IPAddr</td>
<td>IPAddr</td>
<td>-</td>
<td>IP Address of client</td>
<td>-</td>
</tr>
<tr>
<td>Port</td>
<td>Integer</td>
<td>0x0000 – 0xfff</td>
<td>Port of client</td>
<td>0</td>
</tr>
</tbody>
</table>

### 8.3 Device Credentials

There are three credentials per device:

- Short Form Device Identifier (SFDI)
- Long Form Device Identifier (LFDI)
- PIN

8.3.1 Certificate Fingerprint

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

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

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

8.3.2 Short-form Device Identifier (SFDI)

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

Left truncation to 36-bits: 0x3E4F45AB3

Expressed as a decimal: 16726121139

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

For input validation purposes, the sum of the digits of the fingerprint including the checksum digit, modulo 10, SHALL be zero. The SFDI has sufficient entropy (236) to uniquely identify the device in the context of its usage and is used to identify a device within a HAN or site domain. It should not be used in a truly global context (i.e., where the identity of the device cannot be qualified with the domain it is in).

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

8.3.3 Long-form Device Identifier (LFDI)

The LFDI SHALL be the certificate fingerprint left-truncated to 160-bits (20 octets). For display purposes, this SHALL be expressed as 40 hexadecimal (base 16) digits in groups of four. Based on the example in Section 8.3.1, this would be '3E4F-45AB-31ED-FE5B-67E3-43E5-E456-2E31-984E-23E5'.

The LFDI is used when a globally unique identity is required, for example in sending an event back to a service provider that is associated with a particular device.

8.3.4 6-digit PIN code

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

Original PIN: 12345

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

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

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

SFDI || PIN

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

8.4 Resource Access Authentication and Authorization context

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

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

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

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

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

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

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

The application authentication process is as follows:

1) The resource's server listens on the TCP port associated with HTTPS.

2) The client initiates an HTTP request using a random unused source TCP port to the resource's server using the TCP port associated with HTTPS.

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.
b) If the client has a Device Certificate, authentication of the client SHALL be done as part of the TLS handshake by validating the client's Device Certificate as described in [RFC 5246], Section 7 using the inherent PKI. If security policy dictates, additional certificate validation MAY be required. The authentication level to be compared with a resource's corresponding AuthType attribute will be 0x8 (Device Certificate).

c) If the client has a Self-signed Certificate, the Self-signed Certificate SHALL be validated for correctness. The authentication level to be compared with a resource's corresponding AuthType attribute will be 0x4 (Self-signed Certificate).

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

8.6 Resource Access Authorization

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

8.7 Cipher Suites

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

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

8.8 Default Security Policy

Service providers create security policies by balancing the requirements of their regulatory environment and the results of their risk assessments. Different regulatory environments may mandate requirements that trade ease of data access with information assurance. The use of TLS, ACLs, and other security controls give the service provider the flexibility to meet these needs. Security policies are a combination of ACL attribute values and additional security controls dictated by the service provider. Implementation of security policies is out of scope of this specification. For the purpose of certification testing, the following table represents the default security policy for each function set. Servers SHALL be configurable to support each default policy for all implemented function sets during certification testing.

The Function Set column in Table 8-11 reflects the functions Implemented attribute in DeviceInformation.
### Table 8-11 Attribute values for Default Security Policy.

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

The `aclDefaultAccess` attribute Method value SHOULD match the Allowed Methods for each resource enumerated in the SEP 2 WADL [ZB 13-0201]. The Method value MUST contain GET (0x01). The `aclDefaultAccess` attribute DeviceType value should be 'unknown' (0). Servers SHALL support the default policies for certification testing. Servers MAY additionally support alternative policies. For
example, to meet regulatory requirements a utility may mandate a policy that provides unauthenticated pricing information from a pricing server over the port associated with HTTP to any SEP 2 device. Based on risk assessments, service providers may have differing policies for devices enrolled in high-incentive Demand Response / Load Control programs than those enrolled in low-incentive programs, to include additional requirements such as DeviceType authorization. Servers SHOULD provide the functionality to support multiple security policies to meet the requirements of different service providers.

8.9 Registration

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

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

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

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

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

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

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

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

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

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

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

The registration procedure is as follows:
1. The EndDevice resource's server SHALL listen on the TCP port associated with HTTPS and follow the procedure described in [RFC 5246] when a client attempts to access the EndDevice resource.

2. Authorization SHALL then occur whereby the ACLs of the server resources corresponding to the registering client are set according to the security policy and the presence in aclLocalRegistrationList. This ensures that following registration, a client can typically proceed to access all the resources it is authorized to without having to perform any further procedures.

3. If present, the client MUST verify that the EndDevice's associated Registration resource contains the correct PIN for the client. If the PIN does not match, the client SHOULD NOT attempt any further access to that server.

4. The client SHALL subsequently re-use its Device Certificate to authenticate with any other server host. It does not need to re-authenticate with the EndDevice resource's server.

8.10 Security LogEvents

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

Table 8-12 Security LogEvents.

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

8.11 Certificate Management

8.11.1 Introduction

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

In addition, this document describes a few other types of certificates that Smart Energy applications may encounter depending on which services they support – see Section 8.11.8.3 below. While native SEP 2 devices are required to understand, support and process Manufacturing PKI certificates, support of the additional certificates is optional and generally targeted at specific classes of devices (e.g., Energy Services Interface (ESI), web portal).
There are 6 classes of certificates that may be active in a SEP 2 system, depending on configuration and use:

- **Device Certificates** – Issued under the Manufacturing PKI during manufacturing to purpose-built (aka "native") SEP 2 certified devices for operational purposes.
- **Device Test Certificates** – Issued under the Manufacturing PKI during manufacturing to purpose-built (aka "native") SEP 2 certified devices for test purposes.
- **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.
- **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.
- **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.
- **Self-Signed Client Certificate for non-native entities** – A TLS client certificate self-generated and self-signed by a customer or software.

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

- SE2 is Smart Energy (XML) over TLS
- ZIP is Zigbee IP (uses SE2 certs)
- SE2 Cert needed to talk ZIP
- SE2 Manufactured devices ALWAYS have SE2 Cert

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

With the exception of the "AMI Network", the "Internet" and the "Utility", all of the devices pictured above reside in the customer premises. Devices purpose-built for SEP 2 have a manufactured-in Device Certificate. Other devices and entities such as the utilities and a customer-owned tablet computer may
use self-signed certificates or may use certificates issued by commercial CAs. Support for those certificates is mainly dependent on the options implemented by the server.

8.11.2 Certificate Usage – Authentication vs. Authorization

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

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

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

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

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

Table 8-13 TLS Authentication Matrix.

<table>
<thead>
<tr>
<th>Authentication</th>
<th>Server</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Native SEP 2 Application</td>
<td>Generic Server</td>
</tr>
<tr>
<td>Client</td>
<td>Native SEP 2 Application</td>
<td>SEP 2 Cert - Indef</td>
</tr>
<tr>
<td>Generic Client</td>
<td>Optional OCSP</td>
<td>Not Specified Here</td>
</tr>
<tr>
<td>Self-Signed</td>
<td>Signature Validation</td>
<td>Not Specified Here</td>
</tr>
</tbody>
</table>

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

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

Table 8-14 SEP 2 Authorization Matrix.
### 8.11.3 Manufacturing PKI

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

The SEP 2 Manufacturing PKI SHALL be a hierarchy with a depth of 2, 3 or 4 levels. At the top level, Manufacturing PKI hierarchy SHALL have one SERCA. One or more SERCAs SHALL be nominated by the CSEP [CSEP] and SHALL be outsourced to a commercial CA service provider. A SERCA is the property of the CSEP (specifically the private keys associated with the SERCA root are owned by CSEP) and is operated on the CSEP's behalf by a commercial CA. A commercial CA MAY operate one or more SERCAs as business needs dictate and this is one possible model for the initial operation of the SERCAs. A SERCA MAY be moved from one commercial CA to another as business needs dictate.

Private key material for a SERCA MUST be held in a form to allow secure transfer of the material to a new commercial CA if necessary. The SERCA issues MCA certificates and MICA certificates to authorized vendors based on CSEP provided policy. The current version of the specification allows a SEP 2 vendor to contract with a SERCA for the issuance of Device Certificates, but the actual permission to issue such certificates rests with the CSEP.

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

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

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

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

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

- SERCA -> Device Certificate
- SERCA -> MICA -> Device Certificate
- SERCA -> MCA -> MICA -> Device Certificate

8.11.3.1 Manufacturing Certificate Lifecycle

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

An MCA or MICA certificate SHALL NOT be re-issued (e.g., re-signed with the same SubjectName
and public key, but with a new validity period). Instead, if it is desired to retire an existing
key/certificate, a new key pair SHALL be generated and bound into a new MCA or MICA certificate
generally with a new serialNumber component of the SubjectName. Operationally, the responsible CA
SHOULD verifiably\(^1\) destroy the private key of the retired certificate. Retired certificates MUST remain
available to verify subsidiary certificates. A replacement of a MCA or MICA SHOULD use a new name
formed by incrementing or otherwise adjusting the serialNumber component of the subject name. See
Sections 8.11.8.2.1 and 8.11.8.2.2 for the format of the name.

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

8.11.3.2 Device Certificate Lifetime

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

8.11.3.3 Device Certificate Validity

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

8.11.4 General Certificate Format

8.11.4.1 RFC 5280 Compliance

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

8.11.4.2 IEEE 802.1AR Compliance

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

---

\(^1\) The process for destroying private keys is a business issue\(^\dagger\) that should be covered by any contract for SERCA services.
Device Certificates and Device Test Certificates take the general form of an iDevID certificate as defined in [IEEE 802.1AR].

Differing from [IEEE 802.1AR] requirements, the SubjectName field of the Device Certificate is empty as the X500 name form is not well suited to describe or identify physical serialized devices. The device identity is contained in the SubjectAlternativeName extension which contains a single GeneralName of type OtherName that is further sub-typed as a HardwareModuleName (id-on-HardwareModuleName) as defined in [RFC 4108]. Per [RFC 5280], this extension MUST be marked critical when the SubjectName field is empty. The hwType field of HardwareModuleName name is assigned by the SEP 2 manufacturer and SHOULD be different for each different type of manufactured device.

8.11.5 General Restrictions and Conditions

In addition, SEP 2 certificates have the following restrictions:

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

8.11.6 Extensions

- The certificatePolicy extension in any certificate consists of one or more PolicyInformation objects containing only the policyIdentifier field. The PolicyInformation object MUST NOT 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 $\simeq 600K$ certs, 10 octets $\simeq 155M$ certs, 11 octets $\simeq 2.5B$ certs signed without serial number collision.
In the Manufacturing PKI hierarchy, each Device Certificate and the CAs that make up the certificatePath 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. 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 }

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
id=SEP 2-po-service-provider OBJECT IDENTIFIER ::= { SEP 2 Policy 3 }
-- MUST be included in commercial certificates issued to
-- service providers for SEP 2 purposes.
id=SEP 2-po-bulk-cert OBJECT IDENTIFIER ::= { SEP 2 Policy 4 }
-- MUST be included in bulk-issued certificates (e.g.,
-- where the private key is generated off the device by the
-- issuing CA

8.11.7.3 HardwareModuleName

Excerpted from [RFC 4108]:

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

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

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

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

The combination of hwType and hwSerialNum MUST be unique.

Example:

vendor1Devices OBJECT IDENTIFIER ::= { vendor1 13 }
meterNicV1 OBJECT IDENTIFIER ::= { vendor1Devices 1 1 }

HardwareModuleName = {
  OID:1.3.6.1.4.1.99999.13.1.1, -- Vendor1 MeterNic V1
  OCTET STRING: 0x0a43218800 } -- Serial Number 44075-943936

8.11.8 Certificate Profiles

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

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

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

8.11.8.1 Root Certificate

A SERCA instance (e.g., contracted for CA operation) SHALL have exactly one self-signed certificate.
There MAY be more than one SERCA instance.
The SERCA certificates are the root of trust for the Manufacturing PKI. They MUST contain the extensions described below and MUST have the name form as described. They SHOULD NOT contain any additional extensions.

- Issued by: Self-signed
- Issuer Name: O=Smart Energy, CN=SEP 2 Root, serialNumber=<n>
- Subject Name: O=Smart Energy, CN=SEP 2 Root, serialNumber=<n>
- Extensions
  - certificatePolicy: critical; 1:anyPolicy
  - keyUsage: critical; keyCertSign, cRLSign
  - basicConstraints: critical; cA=true, pathLen absent (unlimited)
  - subjectKeyIdentifier: Section 8.11.6

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

### 8.11.8.2 Manufacturing Hierarchy Certificates

#### 8.11.8.2.1 MCA Certificate

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

- Issued by: SERCA
- Subject Name: O=<country>, O=<Manufacturing Org>, CN=SEP 2 MCA, serialNumber=<num>
- Extensions:
  - certificatePolicy: critical; at least one SEP 2 device type Identifier OID as a policyIdentifier
  - keyUsage: critical; keyCertSign
  - basicConstraints: critical; cA=true, pathLen=1
  - subjectKeyIdentifier: Section 8.11.6
  - authorityKeyIdentifier: Section 8.11.6

#### 8.11.8.2.2 MICA Certificate

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

- Issued by: SERCA or MCA
- Subject Name: O=<country>, O=<Manufacturing Org>, CN=SEP 2 MICA, serialNumber=<num>
- Extensions:
  - certificatePolicy: critical; at least one SEP 2 device type Identifier OID as a policyIdentifier
  - keyUsage: critical; keyCertSign
  - basicConstraints: critical; cA=true, pathLen=0
  - subjectKeyIdentifier: Section 8.11.6
  - authorityKeyIdentifier: Section 8.11.6
8.11.8.2.3 **Device Certificate**

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

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

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

8.11.8.2.4 **Device Test Certificate**

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

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

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

8.11.8.3 **Other Certificates**

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

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

8.11.8.3.1.1 **General Considerations**

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

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

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

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

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

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

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

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

The Subject Name form for the Device Certificate MAY be any form approved by the certificate issuer.

This specification recommends that the Subject Name be crafted as: "C=<Country of certification>, O=<Manufacturer>, OU=<Device type>, UID=<Serial Number>". An RSA certificate MUST NOT use a SubjectAlternativeName extension in place of a SubjectName unless it is identical to the SubjectAlternativeName extension in the device's Device Certificate.

8.11.8.3.1.3 Use Case

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

- 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.

- 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.

- 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.

8.11.8.3.2 Self-Signed Client Certificate

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

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

- Issued by: Self-signed
- Subject Name: Any – suggested: O=<application name>, CN=<12digit random hex string>
- Issuer Name: Identical to Subject Name
- Validity: notBefore: time of issue; notAfter: maximum of time of issue plus 3 years.
• Subject Public Key and Signature: SHOULD be EC P-256 and SHA256withECDSA, MAY be RSA 2048 and SHA256withRSA.

• Extensions
  o keyUsage: critical; at least digitalSignature, others as appropriate.
  o certificatePolicy: critical; at least one policyIdentifier:id-SEP 2-po-selfsigned-client.

8.11.8.3.3 Generic Client Certificates for non-SEP 2 Entities

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

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

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

8.11.8.3.4 Generic Server Certificates for non-SEP 2 Entities

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

• MUST contain either an RSA 2048-bit public key or a EC P-256 public key.

• MUST be signed either using either the SHA256withRSA or SHA256withECDSA algorithms.

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

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

8.11.9 Device Requirements

8.11.9.1 Private-key Protection

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

8.11.9.2 Trusted Key Store Protection

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

8.11.9.3 Certificate Usage

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

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

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

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

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

8.11.10  Certificate Verification

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

Certificate verifiers MUST reject certificates that contain one or more unsupported critical extensions. Policy-mapping is not supported, and certificates containing policy mappings MUST be rejected. Policy-qualifiers are not supported. Therefore, issuers SHOULD NOT include policy qualifiers in certificates. However, verifiers SHOULD NOT reject certificates containing policy qualifiers unless there are other reasons to do so. Name-constraints are not supported and certificates containing name-constraints MUST be rejected. Any extension not listed by name within this document SHOULD NOT be included within a compliant certificate and, if included, MUST NOT be marked critical.

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

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

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

8.11.11  Certificate Related Labeling Requirements

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


9 Discovery

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

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

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

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

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

9.1 Service Instance

A server SHALL assign a unique <Instance> label of up to 63-bytes in UTF-8 format for each DNS SRV / TXT record pair that it advertises. In order to avoid name conflicts, <Instance> names SHOULD begin with a meaningful substring followed by a hyphen '-' and end with the device's SFDI or other collision-resistant substring such as the low-order bits of an EUI-64 in text form (e.g., device-000001111114). Should a name conflict occur, a device SHALL assign itself a new name until conflicts
are resolved. A conflict SHOULD be resolved by appending a decimal integer in parentheses to the
Instance> (for example, Name(2) for the first conflict, Name(3) for the second conflict, etc.).

If a DNS SRV/TXT record pair is created to advertise a function set, its Instance SHOULD consist of
the corresponding string from the Subtype column of Table 9-2 followed by a hyphen and a collision-
resistant substring as defined above (e.g., upt-000001111114). When an SFDI is used as part of a DNS-
SD label, it SHALL be represented as 12 decimal digits including leading zeros (if any) as well as the
checksum digit and SHALL NOT include embedded hyphens.

9.2 Service Name

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

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

Thus, a valid Service Instance Name example would be:

device-000001111114._smartenergy._tcp.site.

Where device-000001111114 is the Instance portion (described above), smartenergy is the Service
Name, tcp is the transport protocol, and "site." is the Domain portion (xmDNS is being utilized in
this example).

9.3 TXT Record

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

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

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

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

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

The dcap key SHALL provide the relative reference used to locate the device’s DeviceCapability
resource and SHALL include the leading slash of the given path.
The `dcap` key SHALL be present with a non-empty value. Clients SHALL silently discard TXT records with `dcap` keys that are not present with a non-empty value representing the URI for the server’s DeviceCapability resource.

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

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

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

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

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

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

9.4 Subtype Queries

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

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

A server SHALL register a PTR record with a subtype name as defined below for each function set that it advertises for discovery. In addition, the server SHALL register a unique SRV / TXT record pair with
an <Instance> as defined in Section 9.1 for each function set that is referenced by the subtype PTR record.

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

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

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

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

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

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

To promote search efficiency, servers that support the End Device function set SHALL register a unique PTR, SRV, and TXT record for each remote device that they support. In this case, the subtype name SHALL consist of the string edev, concatenated with a hyphen and the remote device's SFDI, (see Section 8.3.2) and followed by ".smartenergy.tcp.site.". For example, a server having an End Device resource for the remote device with SFDI 222222222228 would register a subtype PTR record named "edev-222222222228.smartenergy.tcp.site.".
The <Instance> portion of the Service Instance Name of the associated SRV / TXT record pair for this subtype PTR SHOULD consist of three parts, concatenated by hyphens. The first part SHALL consist of an identifier unique across End Device instances on this server. The second part SHALL be the edev subtype string. The third part SHOULD be the server's SFDI. For example, a server having the SFDI "000001111114" would register SRV and TXT records having the Service Instance Name "127-edev-000001111114._smartenergy._tcp.site.", where 127 is an arbitrary decimal number unique to this specific End Device instance on this server. This name would be referenced by the subtype PTR record described in the example above. The TXT record with this name would contain the path to the End Device resource of the remote device.

9.5 Discovery Procedure

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

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

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

1) Use xmDNS/DNS-SD to locate the servers with the function sets of interest (See [I-D XMDNS], [RFC 6763]).

2) For each server do the following:
   a) Establish TLS session if required (see Section 8).
   b) GET the Device Capabilities resource (See Section 10.2).
   c) Look for the desired function set in the Device Capabilities resource.
   d) If there is an entry in Device Capabilities it will contain a URI of the entry point for that function set.
   e) Alternately, use the path returned in the subtype query response (See Section 9.4).

3) Determine which of the discovered resources are of interest. This depends on the resource and other outside factors (e.g., if the resource is "metering", then the meter of interest might be the one that has the Premises Aggregation Point attribute set to true).

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

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

10.1 Resource Section Outlines

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

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

1) Overview
2) List Ordering
3) Application Guidelines / Behavior
4) LogEvents

Each of these sections is discussed in more detail below.

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

10.1.2 List Ordering
This section provides guidance regarding; how resource elements and attributes are used to support unique ordering of resources within lists.

Table 10-1 Function Set List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Resource.elementA (descending or ascending)</td>
<td>Resource.elementC (descending or ascending)</td>
<td>NA</td>
</tr>
<tr>
<td>Sub-Resource</td>
<td>Sub-Resource.elementD (descending or ascending)</td>
<td>Sub-Resource.elementB (descending or ascending)</td>
<td>NA</td>
</tr>
</tbody>
</table>

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

This sub-section may also include application guidelines germane to sleepy HAN devices.
10.1.4 **LogEvents**

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

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

Table 10-2 Example LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LE_EXAMPLE_0</td>
<td>0x00</td>
<td>Normative text defining when the event should be generated.</td>
</tr>
<tr>
<td>LE_EXAMPLE_1</td>
<td>0x01</td>
<td>Normative text defining when the event should be generated.</td>
</tr>
</tbody>
</table>

10.2 **Device Capabilities Function Set**

10.2.1 **Overview**

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

10.2.2 **List Ordering**

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

10.2.3 **Application Guidelines / Behavior**

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

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

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

10.2.4 **LogEvents**

There are no LogEvents generated by this function set.

10.3 **Self Device Resource**

10.3.1 **Overview**

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

10.3.2 **List Ordering**

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

10.3.3 **Application Guidelines / Behavior**

If a server hosts resources pointed to from a SelfDevice instance (e.g., DeviceInformation via DeviceInformationLink, PowerStatus via PowerStatusLink), those resources SHOULD be restricted to
read-only access. That is, even if the SEP 2 WADL allows PUT, POST, or DELETE access, those
HTTP methods SHOULD be blocked (or restricted to access by the server itself, i.e. through a loop-back
interface).

Exceptions to the above recommendation are as follows:

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

### 10.3.4 LogEvents

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE_INVALID_WRITE_TO_SELF</td>
<td>0x00</td>
<td>MAY be generated in response to PUT, POST, DELETE attempts on the SelfDevice instance or any of its write-protected sub-resources.</td>
</tr>
</tbody>
</table>

### 10.4 End Device Resource

#### 10.4.1 Overview

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

- EndDeviceList
- EndDevice

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

#### 10.4.2 List Ordering

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndDevice</td>
<td>SFDI</td>
<td>href</td>
<td>N/A</td>
</tr>
</tbody>
</table>

(ascending) (ascending)

#### 10.4.3 Application Guidelines / Behavior

Servers MAY present the EndDeviceList differently based on the credentials of the requesting client.

For example, even if a server contains the EndDevice resources for five registered clients, any one
registered client that GETs the EndDeviceList may receive a list resource containing only the one
EndDevice resource corresponding to that client. Similarly, an unregistered client that GETs the
EndDeviceList may receive an empty EndDeviceList list resource. However, clients SHALL NOT rely
on this filtering behavior, as it is not mandatory server behavior; therefore, clients SHALL support the
normal paging mechanism for List-type resources.
If a server hosts resources pointed to from an EndDevice instance (e.g., DeviceInformation, DevicePowerStatus) that allow PUT, POST, or DELETE access, those HTTP methods SHOULD be restricted to the client device represented by the given EndDevice.

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

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

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

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

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

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

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

### 10.4.4 LogEvents

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LE_UNAUTHORIZED_WRITE_TO_END_DEVICE</td>
<td>0x00</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 10.5 Function Set Assignments

#### 10.5.1 Overview

The FunctionSetAssignments resource defines collections of references to function set instances. These collections are used by the End Device resource for indicating to devices the resources that are to be used for specific purposes. For example, a service provider may wish that all the participants of a
particular program use a given DRLC function set instance, Time resource, and Pricing function set instance. A FunctionSetAssignments resource contains references to the function set instances that are to be used by the assigned device(s).

10.5.2 List Ordering

Table 10-6 Function Set Assignments List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunctionSetAssignments</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

10.5.3 Application Guidelines / Behavior

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

Specific instances of FunctionSetAssignments resources are defined out of band and the methods for getting the information into the instances is outside the scope of this specification. Server use of this function set is OPTIONAL. Clients that act on events SHALL determine if they are assigned into Function Set Assignments. Clients MAY be assigned multiple Function Set Assignments. Multiple Clients MAY be assigned the same Function Set Assignment. If a server supports Function Set Assignments it SHALL support a minimum of 1 Function Set Assignments for each HAN device registered to the server. A server SHOULD support 3 Function Set Assignments for each HAN device.

Clients SHALL support at least 6 function set instances (e.g., two DemandResponseProgram instances, one Time instance, one TariffProfile instance, and two MessagingProgram instances) assigned through 1 or more Function Set Assignments.

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

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

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

10.5.4 LogEvents

There are no LogEvents generated by this function set.

10.6 Subscription / Notification Mechanism

10.6.1 Overview

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

The following text further clarifies the roles of the subscription and notification resources.
10.6.2 List Ordering

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

Table 10-7 Subscription / Notification List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription</td>
<td>href (ascending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Notification</td>
<td>href (ascending)</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

10.6.3 Application Guidelines / Behavior

10.6.3.1 Subscription Resource

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

10.6.3.2 Notification Resource

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

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

10.6.3.3 Conditional Subscription / Report by Exception

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

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

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

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

For a given resource, to determine the attribute on which the conditions apply, the attributeIdentifier attribute is used.
10.6.3.4 Subscription Rules

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

1) Clients SHOULD send subscription renewals every 24 hours, and no more often than every 1 hour, to sustain a subscription.

2) Servers MAY remove subscriptions at any time.

3) Servers SHOULD remove subscriptions if a client has not renewed in 36 hours.

4) Subscriptions SHALL be maintained on servers through power loss.

5) Servers SHALL use the subscription parameters from the latest subscription renewal.

6) Clients SHOULD poll after perceived loss of connectivity.

7) If the URI of a resource changes, then subscriptions to that resource SHALL be terminated.

8) For subscriptions to non-list resources, a notification SHALL be sent whenever the representation of the non-list resource changes.

9) For subscriptions to list resources, a notification SHALL be sent whenever any subordinate resources are added to or removed from the list, or if the representation of those subordinate resources change. As a result, if a client is subscribed to both an event and a list containing that event, the client will receive two notifications should that event change. See Section 6.7 for more information.

   a. It should be noted that notifications for list resources are an exception where certain list attributes are included (e.g., all and results) that would normally not be present when a list resource is provided in a POST.

10) To prevent overwhelming network resources, notifications SHOULD be sent to a given client for a given resource no more than once every 30 seconds. Notifications for conditional subscriptions SHOULD only be sent once within this time period for a given client for a given resource and any additional notifications SHOULD NOT be queued. All devices need to be considerate of network resources.

11) Servers implementing the subscription resource SHALL be capable of maintaining a minimum of 1 subscription and servers implementing the subscription resource SHOULD support at least 1 subscription per device per subscribable resource.

12) If a server implementing the subscription resource is unable to accept a subscription, the server SHALL return an error resource representation indicating the specific error (e.g., element does not support conditional subscription) with an HTTP response code of 400.

13) When a server removes or terminates a subscription, it SHALL send the client a terminate subscription (except after an error from an undesired subscription, as mentioned in rule 14 below). The server SHALL send a Notification to the client's "Post URI" for the affected subscription. The Resource contained in the Notification SHALL be a Subscription, containing a Status element identifying the reason for terminating the subscription. When a subscription is terminated because the subscribed resource has moved, servers MAY include newResourceURI in the subscription termination message, indicating the resource’s new location.

14) If a client receives a notification for an undesired subscription, the client SHALL return an HTTP 400 error. Upon receipt of such an error, the server SHALL remove the subscription without notification.

15) Servers SHOULD allow only the end device that corresponds to a given EndDevice resource to modify the subscriptions within that resource.
16) The default recommended policy is that subscription management SHOULD be performed using TLS.

10.6.4 LogEvents

Table 10-8 Subscription / Notification LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB_NTFY_FAIL</td>
<td>0x00</td>
<td>SHOULD be issued when there is a failure to successfully send a Notification (after a successful connection).</td>
</tr>
<tr>
<td>SUB_CONN_ESTB_FAIL</td>
<td>0x01</td>
<td>SHOULD be issued when there is a failure to successfully establish a connection to send a Notification.</td>
</tr>
</tbody>
</table>

10.7 Response

10.7.1 Overview

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

The originating event will indicate if a response is required.

10.7.2 List Ordering

Table 10-9 Response List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResponseSet</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Response</td>
<td>createdDateTime</td>
<td>endDeviceLFDI</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>(descending)</td>
<td>(ascending)</td>
<td></td>
</tr>
</tbody>
</table>

10.7.3 Application Guidelines / Behavior

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

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

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

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

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

The Response Server identified in the replyTo field of the event SHOULD be available on the network.
Several function sets use the `responseRequired` attribute, which indicates whether or not this particular event requires a response from the client. The following table indicates the valid `Response.type` values supported by each function set:

Table 10-10 Response Types by Function Set.

<table>
<thead>
<tr>
<th>Enumeration Value</th>
<th>Description</th>
<th>Why Sent</th>
<th>When Sent</th>
<th><code>responseRequired</code> bit position</th>
<th>DRLC and DER</th>
<th>Pricing</th>
<th>Messaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>Event received</td>
<td>To notify response server that client has initially received event.</td>
<td>When the device first receives the event, either via a GET or a Notification.</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Event started</td>
<td>To notify response server that client has begun event.</td>
<td>At <code>EffectiveStartTime</code> (see Section 12.1.3.1)</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Event completed</td>
<td>To notify response server that the event fully and successfully completed.</td>
<td>At <code>EffectiveEndTime</code> (see Section 12.1.3.1)</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>User has chosen to opt-out</td>
<td>To notify response server that user has chosen to opt out of the event. Can occur before event begins.</td>
<td>At time user actively chose to opt out or when device automatically opts out due to user preference.</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User has chosen to opt-in</td>
<td>To notify response server that user has chosen to opt in to the event (after a previous opt-out). Can occur before event begins.</td>
<td>At time user actively chose to opt in or when device automatically opts in due to user preference.</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>The event has been cancelled</td>
<td>To notify response server that client has initially received cancellation.</td>
<td>When the device first receives the cancellation, either via a GET or a Notification, even if the event has not yet begun execution.</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Enumeration Value</td>
<td>Description</td>
<td>Why Sent</td>
<td>When Sent</td>
<td>responseRequired</td>
<td>DRLC and DER</td>
<td>Pricing</td>
<td>Messaging</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>7</td>
<td>The event has been superseded</td>
<td>To notify response server that client has learned of an event that</td>
<td>When the device first learns of an event that supersedes this event,</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>supersedes this event.</td>
<td>even if the event has not yet begun execution.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Event partially completed with user opt-out</td>
<td>To notify response server that some participation in the event occurred,</td>
<td>At EffectiveEndTime (see Section 12.1.3.1)</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>but not complete participation, due to one or more user opt-outs.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Event partially completed due to user opt-in</td>
<td>To notify response server that some participation in the event occurred,</td>
<td>At EffectiveEndTime (see Section 12.1.3.1)</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>but not complete participation, due to one or more user opt-ins (after a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>previous opt-out).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Event completed, no user participation (previous</td>
<td>To notify response server that no participation in the event occurred.</td>
<td>At EffectiveEndTime (see Section 12.1.3.1)</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>opt-out)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>User has acknowledged the event</td>
<td>To notify response server that the client displayed the event and</td>
<td>At time user actively chose to acknowledge the event.</td>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a human user actively acknowledged it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Cannot be displayed</td>
<td>To notify response server that the client is unable to display the</td>
<td>When the device first receives the event, either via a GET or a</td>
<td>1</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>message.</td>
<td>Notification.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enumeration Value</td>
<td>Description</td>
<td>Why Sent</td>
<td>When Sent</td>
<td>responseRequired</td>
<td>DRLC and DER</td>
<td>Pricing</td>
<td>Messaging</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------</td>
<td>--------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>13</td>
<td>Event aborted due to alternate provider event</td>
<td>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.</td>
<td>At time the original event was ceased.</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>14 - 251</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>252</td>
<td>Rejected – control parameter not applicable</td>
<td>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).</td>
<td>When the device first receives the event, either via a GET or a notification.</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>253</td>
<td>Rejected – invalid event</td>
<td>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).</td>
<td>When the device first receives the event, either via a GET or a notification.</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>254</td>
<td>Rejected – event was received after it expired</td>
<td>To notify response server that client has initially received event, but that the event was received after <code>SpecifiedEndTime</code>.</td>
<td>When the device first receives the event, either via a GET or a Notification.</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10.7.3.1 **Response Storage**

It is expected that the server provides some mechanism that allows the service provider to obtain the responses, even in the presence of outages. Details of storage are vendor specific.
Implementers SHOULD have enough storage and bandwidth to support responses for the number of
devices they plan to support as a server for each function set that requires a response.

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

10.7.4 LogEvents

Table 10-11 Response LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE_LIST_OVERFLOW</td>
<td>0x00</td>
<td>Should be issued by a server anytime their response list has exceeded maximum storage.</td>
</tr>
<tr>
<td>RESPONSES_REPORTED_UPSTREAM</td>
<td>0x01</td>
<td>Should be issued by a server when responses are sent upstream.</td>
</tr>
</tbody>
</table>
11 Common Resources

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

11.1 Time Function Set

11.1.1 Overview

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

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

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

11.1.2 List Ordering

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

11.1.3 Application Guidelines / Behavior

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

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

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

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

If a device is not assigned a Time resource, it MAY discover and operate Event-based function sets.

When the device executes these events, the device SHALL use the Time resource from the server device from which it received the event. If a client discovers an Event-based function set and cannot also retrieve a Time resource from this same server, it SHALL NOT act on the events from this Event-based function set.

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

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

Some service providers operate a given asset on a time standard that differs from the definition given for TimeType. This would be an intentionally uncoordinated time. If a time server serves an intentionally
uncoordinated time in the currentTime field, it SHALL have a quality metric of 7 - time intentionally uncoordinated.

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

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

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

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

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

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

Time related configuration is handled within the Configuration resource.

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

11.1.4 LogEvents

Table 11-1 Time LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM_TIME_ADJUSTED</td>
<td>0x00</td>
<td>Should be issued by a server when its time is adjusted.</td>
</tr>
<tr>
<td>TM_TIME_SOURCE_CHANGED</td>
<td>0x01</td>
<td>Should be issued by a client when it chooses a different primary time source.</td>
</tr>
</tbody>
</table>

11.2 DeviceInformation Function Set

11.2.1 Overview

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

11.2.2 List Ordering

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

11.2.3 Application Guidelines / Behavior

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

11.2.4 LogEvents

There are no LogEvents generated by this function set.
11.3 **Power Status**

11.3.1 **Overview**

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

11.3.2 **List Ordering**

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

11.3.3 **Application Guidelines / Behavior**

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

11.3.4 **LogEvents**

**Table 11-2 Power Status LogEvents.**

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS_LOW_BATTERY</td>
<td>0x00</td>
<td>Should be issued when the installed battery charge drops below the configured low battery charge threshold.</td>
</tr>
</tbody>
</table>

11.4 **Network Status**

11.4.1 **Overview**

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

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

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

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

11.4.2 **List Ordering**

**Table 11-3 Network Status List Ordering.**

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPInterface</td>
<td>ifIndex (ascending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>IPAddr</td>
<td>address (ascending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>LLInterface</td>
<td>EUI64 (ascending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>RPLInstance</td>
<td>DODAGid (ascending)</td>
<td>RPLInstanceID (ascending)</td>
<td>NA</td>
</tr>
</tbody>
</table>
11.4.3 Application Guidelines / Behavior

This function set does not have any specific application guidelines / behavior beyond those already specified in the schema.

11.4.4 LogEvents

Table 11-4 Network Status LogEvents

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

11.5 LogEvent List

11.5.1 Overview

The LogEvent List contains a list of time-stamped instances of LogEvents generated by the device. LogEvents are typically asynchronously generated warnings of some out of bounds condition or unusual significant event detected by the device. For example, the device's battery charge falling below an operator designated low charge threshold.

11.5.2 List Ordering

Table 11-5 LogEvent List Ordering.
11.5.3 **Application Guidelines / Behavior**

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

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

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

11.5.3.1 **LogEvents**

There are no LogEvents generated by this function set.

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

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LE_GEN_HARDWARE</td>
<td>0x00</td>
<td>Unclassified general hardware fault occurred.</td>
</tr>
<tr>
<td>LE_GEN_SOFTWARE</td>
<td>0x01</td>
<td>Unclassified general software fault occurred.</td>
</tr>
</tbody>
</table>

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

11.6 **Configuration Resource**

11.6.1 **Overview**

The Configuration resource implements centralized read / write access to the device's operational configuration. This resource may be queried to provide current device configuration, perform configuration backup, etc. This resource is modified as needed to control the operation of the device. This resource adheres to the recommendations outlined in [RFC 3535]. Specifically, configuration is best separated from operational state and performance statistics to ease configuration retrieval, conformance check, and placement / modification to the device. In other words, it is easier to manage device configuration via a single resource than having to collect, collate, and update configuration parameters from multiple resources spread across a SEP 2 implementation.

11.6.2 **List Ordering**

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

11.6.3 **Application Guidelines / Behavior**

It is strongly RECOMMENDED that devices persist configuration data across a power cycle.
### 11.6.4 LogEvents

Table 11-7 Configuration LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFG_CONFIGURATION_UPDATEd</td>
<td>0x00</td>
<td>SHALL be issued when the local Configuration resource is changed.</td>
</tr>
</tbody>
</table>

### 11.7 File Download Function Set

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

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

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

### 11.7.1 File List Resource

#### 11.7.1.1 Overview

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

#### 11.7.1.2 List Ordering

File resources are ordered within a FileList as follows:

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
<th>Quaternary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>mflID</td>
<td>mfModel</td>
<td>mfVer</td>
<td>href</td>
</tr>
<tr>
<td></td>
<td>(ascending)</td>
<td>(ascending)</td>
<td>(descending)</td>
<td>(ascending)</td>
</tr>
</tbody>
</table>

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

#### 11.7.1.3 Application Guidelines / Behavior

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

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

The LD issues discovery requests to locate available FS (DNS-SD subtype "file"). The LD MUST first determine if there exists a FunctionSetAssignment, containing a FileList, for the device. If such a FileList exists, the LD MUST use this FileList exclusively. If a Time resource is specified within this same FunctionSetAssignment, the LD MUST use this Time resource as the reference for file activation.
If the LD does not locate a FunctionSetAssignment directing the LD to a specific FileList, the LD MUST attempt discovery of a FileList provided by any device's DeviceCapabilities resource.

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

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

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

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

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

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

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

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

An LD SHOULD implement handling for 503 errors and SHOULD implement handling of the Retry-After entity header. If the LD does not support processing of the Retry-After entity header, the LD MUST wait at least 30 seconds before retrying the file content request.
An LD SHOULD negotiate a desired TCP window size with the FS.

Files SHOULD be treated as raw binary, thus a Content-Type entity header of "application/octet-stream" SHOULD be used.

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

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

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

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

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

1) type
2) IFDI
3) mfHwVer
4) mfID
5) mfModel
6) mfSerNum
7) mfVer

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

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

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

The set of file query parameters MUST be interpreted as a Boolean expression, with each file query parameter considered a clause of that expression, and each clause connected by a logical AND operand.
Each file query parameter MUST provide an exact match with its File equivalent for its clause to evaluate to true. There is one exception to this case: the mfVer MUST be evaluated as a GREATER THAN logical operand versus its File equivalent.

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

11.7.1.4 LogEvents

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

Table 11-9 File Download LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE_FILE_LOAD_FAILED</td>
<td>0x00</td>
<td>This event SHALL be recorded by the LD when it is unable to load a file that was indicated as available by the FS.</td>
</tr>
<tr>
<td>FE_FILE_LOAD_OK</td>
<td>0x01</td>
<td>This event SHALL be recorded by the LD when a file is successfully loaded to the LD.</td>
</tr>
<tr>
<td>FE_FILE_SIGNATURE_VERIFICATION_FAILED</td>
<td>0x02</td>
<td>This event SHALL be recorded by the LD when it is unable to verify the signature of a loaded file.</td>
</tr>
<tr>
<td>FE_FILE_SIGNATURE_VERIFICATION_OK</td>
<td>0x03</td>
<td>This event SHALL be recorded by the LD when a loaded file's signature is successfully verified.</td>
</tr>
<tr>
<td>FE_FILE_ACTIVATION_FAILED</td>
<td>0x04</td>
<td>This event SHALL be recorded by the LD when it is unable to activate a previously loaded and signature verified file.</td>
</tr>
<tr>
<td>FE_FILE_ACTIVATION_OK</td>
<td>0x05</td>
<td>This event SHALL be recorded when the LD has successfully activated a loaded and signature verified file.</td>
</tr>
<tr>
<td>FE_FILE_MULTIPLE_FILES</td>
<td>0x06</td>
<td>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.</td>
</tr>
</tbody>
</table>

11.7.2 File Status Resource

11.7.2.1 Overview

The FileStatus resource enables LDs to provide a means to track the progress of any file load operations and also a means to diagnose failed file load operations.
11.7.2.2 **List Ordering**

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

11.7.2.3 **Application Guidelines / Behavior**

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

11.7.2.4 **LogEvents**

There are no LogEvents generated by this resource.
12 Smart Energy Resources

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

12.1 Common Functionality

12.1.1 Overview

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

12.1.2 Active List Elements

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

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

12.1.3 Event Rules and Guidelines

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

12.1.3.1 Definitions

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

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

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

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

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

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

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

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

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

Duration Randomization – The bound on the amount of time to be used when randomizing the completion of an Event.

Effective Duration - Represents Duration with Duration Randomization applied.

Overlapping Event - Defined as an Event where the Scheduled Period covers part or all of an existing, previously scheduled Event.

Start Randomization – The bound on the amount of time to be used when randomizing the commencement of an Event.

Successive Events - When the Start Time plus Duration of the first event is the same as the Start Time of the second event without randomization.

Nested Events - Defined as two Events where the scheduled Start Time and Specified End Time of the second Event falls during the Scheduled Period of the first Scheduled Event and the second Event is of shorter Duration than the first Event. This is an extreme case of Overlapping Event.

12.1.3.2 Rules and Guidelines

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

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

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.

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.

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.

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.

5) A client SHALL consider the current Event complete if a superseding Event is started.

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.
7) Events presented to the HAN SHOULD make minimum use of Overlapping Events and Nested Events. Overlapping Events and Nested Events SHOULD only be used where changing conditions mandate superseding previous Events.

8) When changing conditions mandate changes in the sequence or contents of Events, the following guidelines MAY be used to indicate desired actions:
   a. Canceling existing Events and reissuing new Events.
   b. Sending overlapping or nested Events to supersede existing Events.

9) When a Nested Event completes the containing / superseded Event SHALL NOT be reinstated and remain in a superseded state.

10) For function sets with direct control, it is RECOMMENDED that process 8.b be used for most situations since it can allow a smoother change between two sets of directives but in no way does it negate the responsibilities identified in rule #7.

11) Clients SHALL verify the EventStatus of an Event before acting upon it. If the EventStatus potentiallySupersededTime has changed since last checked, and if the EventStatus type is "Partially Superseded", clients SHALL check all Events from that function set instance that may supersede the original Event.

12) When a client receives an Event with the Specified End Time in the past (Specified End Time < Current Time), this Event SHALL be ignored. Note that the Duration Randomization is not used in this calculation.
   a. For function sets with direct control, if the Event responseRequired indicates, clients SHALL POST a Response to the replyTo URI with a Status of "Rejected - Event was received after it had expired".

13) When a client receives an Event and calculates an Effective Start Time (Start Time + Start Randomization) in the past and a Specified End Time in the future ((Effective Start Time < Current Time) AND (Specified End Time > Current Time)), the client SHALL begin the Event using the current time and the absolute value of Start Randomization. For response reporting purposes, the start time SHALL be reported as the Current Time plus applied Start Randomization applied. For Event duration purposes, the Specified End Time SHALL be preserved, and any Duration Randomization attributes SHALL be applied to the abbreviated Duration.

14) For function sets with direct control, regardless of the state of an Event (scheduled or active), when a client detects an Overlapping Event condition, the Event with the latest creation time will take precedence over the previous Event. Depending on the state of the Event (scheduled or active), one of the following steps SHALL take place:
   a. If the previous Event is scheduled and not active and if the Event responseRequired indicates, the client SHALL POST a Response (referencing the previous Event) with the Status of "The Event has been superseded." After the Response has been successfully POSTed, the client SHALL ignore the previous Event scheduled.
   b. If the previous Event is active, the client SHALL change directly from its current state to the requested state at the effective start time of the Overlapping Event. If the Event responseRequired indicates, the client SHALL POST a response (referencing the previous Event) with a Status of 'The event has been superseded' at the effective start time of the Overlapping Event.
15) Randomization SHALL NOT cause Event conflicts or unmanaged gaps. To clarify:
   a. For Successive Events clients SHALL use the earlier Event’s Effective End Time as the 
      Effective Start Time of the later Event. Events are not reported as superseded and Clients 
      should report Event statuses as they normally would for a set of Successive Events.
      Note: This means that a group of Successive Events without Duration Randomization 
      will run successively using the initial Start Randomization for each of the Events in the 
      group.
   b. Randomization SHALL NOT artificially create a gap between Successive Events.

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

17) If multiple EndDeviceCategoryTypes are identified for an Event, future Events for an individual 
    EndDeviceCategoryType (or a subset of the original Event) that cause an Overlapping Event 
    will supersede the original Event strictly for that EndDeviceCategoryType (or a subset of the 
    original Event). Note: Rule #6 applies to all Overlapping Events.
    a. Those clients whose EndDeviceCategoryType is not listed in the future Event but whose 
       EndDeviceCategoryType was included in the original Event SHALL continue to 
       execute per the parameters of the original Event.
    b. Rule #3 continues to apply. Servers SHALL NOT edit the original Event but SHALL 
       maintain all Events in their entirety.
    c. A server SHALL set the potentiallySuperseded flag when the Event is superseded for 
       any of the device categories and update the potentiallySupersededTime.

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

19) When an Event is removed from the server (e.g., due to limited storage space for the Event list) 
    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.

12.1.4 Randomization

12.1.4.1 Overview

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

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

12.1.4.2 Randomization Attributes

Randomization consists of two signed attributes: randomizeStart and randomizeDuration. Neither, one, 
or both of these may be included as part of an Event.
Randomization is implemented using signed integer(s) to indicate whether a HAN device executes a control action before or after the start or duration (or both start and duration) of the related Event. Time granularity for randomization is one second.

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

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

12.1.4.2.1 randomizeStart

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

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

12.1.4.2.2 randomizeDuration

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

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

12.1.4.3 Application Guidelines / Behavior

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

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

Cancellation of an active Event SHALL cause clients to apply the greater of (absolute value of randomizeStart attribute) and (absolute value of randomizeDuration) to the abbreviated Event.
If an Event is in progress and user override occurs, the client SHALL respond to the user override without randomization.

12.1.5 Multi-Server

12.1.5.1 Overview

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

12.1.5.2 Registration

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

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

12.1.5.3 Time

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

12.1.5.4 Messaging Function Set

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

12.1.5.5 Pricing Function Set

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

12.1.5.6 DRLC and DER Control Function

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

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

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

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

2) Clients SHALL prioritize execution of DRLC and DER function set Events with different PrimacyType attributes using the following guidelines:
   - 0 supersedes 1
   - 1 supersedes 2
   - 2 supersedes 3
   - If two instances are received with the same priority, then normal Event Rules and Guidelines apply (e.g., superseding based on scheduling).

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

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

12.2 Demand Response and Load Control

12.2.1 Overview

This function set provides an interface for Demand Response and Load Control (DRLC). Client devices of this function set include thermostats and devices that support load control. Server devices of this function set include ESIs, premises energy management systems that may be acting as a proxy for upstream demand response / load control management systems and subsequent data stores. Servers expose load control events to client devices through resources called "EndDeviceControls" (EDC). All EDC instances expose attributes that allow devices to respond to events that are explicitly targeted at their device type. For example, an EDC may contain an Offset object indicating a degree offset to be applied by a Smart Thermostat. The EDC will also expose necessary attributes that load control client devices will need in order to process an event. These include Start Time and Duration, as well as an indication on the need for randomization of the start time or duration of the event.

12.2.2 List Ordering

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

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 12.2.3 Application Guidelines / Behavior

#### 12.2.3.1 DemandResponseProgram

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

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

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

#### 12.2.3.2 EndDeviceControl

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

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

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

Demand Response / Load Control client devices SHALL be capable of internally storing and supporting at least 3 unique EndDeviceControl instances. Demand Response / Load Control client devices SHOULD be capable of internally storing and supporting at least 5 unique EndDeviceControl instances.

As a highly RECOMMENDED recovery mechanism, when a maximum of storage of events has been reached and additional EndDeviceControl instances are available on the server(s), clients SHOULD prioritize local storage and give preference to EndDeviceControls with start times in the near future and to events that have been flagged as mandatory.

#### 12.2.3.3 Rules and Guidelines

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

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

#### 12.2.3.3.1 availabilityUpdateChangePercentThreshold/ availabilityUpdateChangePowerThreshold

When Demand Response / Load Control servers support the LoadShedAvailability resource, they SHALL use either the availabilityUpdateChangePercentThreshold or availabilityUpdateChangePowerThreshold to indicate the threshold for which a client is required to update its current load shed ability. The availabilityUpdateChangePercentThreshold is used to indicate a
percent change in load shed availability that SHALL trigger an update from the client. The
availabilityUpdateChangePowerThreshold is used to indicate an absolute change in available load shed
capability that SHALL trigger an update from the client. If no
availabilityUpdateChangePercentThreshold or availabilityUpdateChangePowerThreshold is specified
then clients SHALL NOT POST to their LoadShedAvailability for that program. If the server provides
both, the client SHALL update its current load shed availability when either threshold is crossed.

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

12.2.3.3.3 overrideDuration
After the overrideDuration time of an EndDeviceControl has elapsed, the client device SHALL return to
execution of the EndDeviceControl for the remaining Effective Scheduled Period.
Client devices MAY allow users to override an EndDeviceControl for a longer duration than event
overrideDuration, in which case they SHOULD provide a warning for non-compliance if the
drProgramMandatory flag is set to true.

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

12.2.3.3.5 SetPoint
When both a Temperature Offset and a Temperature Set Point are provided, the device MAY use either
as defined by the device manufacturer.
In some cases an EDC MAY include an Offset or Set Point that will cause the device to increase its
energy consumption. This may be done as a form of pre-heating or pre-cooling before an event that
reduces consumption is dispatched. This type of EDC SHALL set the loadShiftForward flag to "True".

12.2.3.4 Response
When overriding an event, client devices SHOULD provide a duration for the override using the
drOverrideDuration attribute found in the DrResponse object. This is useful for service providers and
EMS in understanding for how long the client device will override the event and when it can expect the
client device to return to shedding load.
Additional guidelines on how the Response resource operates are provided in the Response Function Set
in Section 10.7.

12.2.3.5 LoadShedAvailability
When clients support the LoadShedAvailability resource they SHALL POST their ability to shed load
when first connected to a server that supports the resource. If the commitment is related to a program,
the client SHALL provide a link to the specific DemandResponseProgram. Clients SHALL update their
availability based on the update thresholds provided by the program where the load shed is being
committed. If no thresholds are specified by the server, then clients SHALL NOT POST updates to their
LoadShedAvailability for the provider of that program.
Clients reporting their LoadShedAvailability to multiple DemandResponsePrograms SHALL be capable of individually providing the committed power or percentage to each program for a reported duration. For example, a client capable of shedding 2kW load SHALL NOT report 2kW of availability to two separate programs, instead it SHALL divide its availability between the two. Client devices already shedding load and reporting on additional availability, SHALL be aware of their current state of consumption, as any new event requiring energy reduction SHALL be applied to the device's current baseline of consumption.

12.2.4 LogEvents
There are no LogEvents generated by this function set.

12.3 Metering Function Set

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

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

12.3.2 List Ordering

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>UsagePoint</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MeterReading</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>ReadingSet</td>
<td>timePeriod.start (descending)</td>
<td>mRID (descending)</td>
<td>NA</td>
</tr>
<tr>
<td>Reading</td>
<td>localID (ascending)</td>
<td>consumptionBlock (ascending)</td>
<td>touTier (ascending)</td>
</tr>
</tbody>
</table>

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

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

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

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

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

- SHALL contain a ReadingTypeLink element that points to a ReadingType resource that matches the SummationDelivered definition from Table 12-3.
  - The ReadingType resource SHALL specify the number of TOU tiers and / or consumption blocks, if any, that the metering instance provides.
- SHALL contain a ReadingSetListLink element that points to a ReadingSetList resource.
  - When metering data is present, the ReadingSetList SHALL contain at least one Reading resource, which corresponds to the present summation delivered data.
    - The ReadingSet resource SHALL contain a ReadingListLink element that points to a ReadingList resource. The ReadingList resource SHALL contain (number of TOU tiers + 1) multiplied by (number of consumption blocks + 1) Reading resources.
    - The Reading resources in the ReadingList SHALL correspond to the summation delivered value for each combination of consumptionBlock=0..(number of consumption blocks) and touTier=0..(number of TOU tiers).
    - The Reading for (consumptionBlock=0, touTier=0) SHALL correspond to the total summation delivered.
    - The Reading for (consumptionBlock=x > 0, touTier=0) SHALL correspond to the Block x summation delivered (across all TOU tiers).
    - The Reading for (consumptionBlock=0, touTier=y > 0) SHALL correspond to the TOU Tier y summation delivered (across all consumption blocks).
    - The Reading for (consumptionBlock=x > 0, touTier=y > 0) SHALL correspond to the consumptionBlock=x, touTier=y summation delivered.
- SHALL contain a ReadingLink to a Reading resource that contains the present summation delivered, which is semantically equivalent to the Reading for (consumptionBlock=0, touTier=0) for the present ReadingList.
A Metering function set instance that provides summation received data, maximum demand delivered data, maximum demand received data, and / or other reading type data that utilizes TOU tiers and / or consumption blocks SHALL serve a Meter Reading resource (and subordinate resources) as per the rules for Summation Delivered above, but with the ReadingType resource matching the appropriate definition in Table 12-3, and with the Reading values corresponding to the appropriate data type.

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

- SHALL contain a ReadingTypeLink element that points to a ReadingType resource that matches an Interval data definition from Table 12-3.
  - The ReadingType resource SHALL specify the intervalLength that is the default for the intervals contained in the ReadingList resource.
- SHALL contain a ReadingSetListLink element that points to a ReadingSetList resource.
  - When metering data is present, the ReadingSetList SHALL contain at least one ReadingSet resource, which corresponds to the present Interval Block data (the one currently being filled).
    - The ReadingSet resource SHALL contain a ReadingListLink element that points to a ReadingList resource. The ReadingList resource SHALL contain Reading resources each of which represents a portion (interval) of the timePeriod of the ReadingSet.
    - If the duration in the Time Period of the Reading is not equal to the intervalLength specified in the ReadingType the timePeriod SHALL be included in the Reading.
- SHALL NOT contain a ReadingLink resource.
The following table provides a list of common Reading Type Definitions with related fields listed.

**Table 12-3 Reading Types.**

<table>
<thead>
<tr>
<th>ReadingType Element</th>
<th>accumulationBehaviour</th>
<th>calorificValue</th>
<th>commodity</th>
<th>conversionFactor</th>
<th>dataQualifier</th>
<th>flowDirection</th>
<th>intervalLength</th>
<th>kind</th>
<th>numberOfConsumptionBlocks</th>
<th>numberOfTiers</th>
<th>phase</th>
<th>powerOfTenMultiplier</th>
<th>subIntervalLength</th>
<th>supplyLimit</th>
<th>tieredConsumptionBlocks</th>
<th>uom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantaneous Demand</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>E</td>
<td>8</td>
<td>O</td>
<td>E</td>
<td>E</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summation Delivered</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>E</td>
<td>12</td>
<td>O</td>
<td>O</td>
<td>E</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summation Received</td>
<td>9</td>
<td>1</td>
<td>19</td>
<td>E</td>
<td>12</td>
<td>O</td>
<td>O</td>
<td>E</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max Demand Delivered</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>O</td>
<td>8</td>
<td>O</td>
<td>O</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max Demand Received</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>19</td>
<td>O</td>
<td>8</td>
<td>O</td>
<td>O</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervals Delivered</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>O</td>
<td>12</td>
<td></td>
<td></td>
<td>O</td>
<td>E</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervals Received</td>
<td>4</td>
<td>1</td>
<td>19</td>
<td>O</td>
<td>12</td>
<td></td>
<td></td>
<td>O</td>
<td>E</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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].
12.3.4 **LogEvents**  
This subsection includes definitions of all LogEvents that may be raised by the metering function set.  
The LogEvent names and codes are summarized in the table below.  

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPT_CHECK_METER</td>
<td>0x00</td>
<td>SHOULD be issued when check meter alarm occurs</td>
</tr>
<tr>
<td>UPT_TAMPER_DETECT</td>
<td>0x01</td>
<td>SHOULD be issued when a tampering is detected</td>
</tr>
<tr>
<td>UPT_POWER_QUALITY</td>
<td>0x02</td>
<td>SHOULD be issued when power quality alarm occurs. It is a generic power quality event code</td>
</tr>
<tr>
<td>UPT_LEAK_DETECT</td>
<td>0x03</td>
<td>SHOULD be issued when a leak is detected</td>
</tr>
<tr>
<td>UPT_SERVICE_DISCONNECT</td>
<td>0x04</td>
<td>SHOULD be issued when service is disconnected</td>
</tr>
<tr>
<td>UPT_SERVICE_LIMITED</td>
<td>0x05</td>
<td>SHOULD be issued when service limited alarm occurs</td>
</tr>
<tr>
<td>UPT_LOW_VOLTAGE_L1</td>
<td>0x06</td>
<td>SHOULD be issued when low voltage L1 occurs</td>
</tr>
<tr>
<td>UPT_HIGH_VOLTAGE_L1</td>
<td>0x07</td>
<td>SHOULD be issued when high voltage L1 occurs</td>
</tr>
<tr>
<td>UPT_LOW_VOLTAGE_L2</td>
<td>0x08</td>
<td>SHOULD be issued when low voltage L2 occurs</td>
</tr>
<tr>
<td>UPT_HIGH_VOLTAGE_L2</td>
<td>0x09</td>
<td>SHOULD be issued when high voltage L2 occurs</td>
</tr>
<tr>
<td>UPT_LOW_VOLTAGE_L3</td>
<td>0x0A</td>
<td>SHOULD be issued when low voltage L3 occurs</td>
</tr>
<tr>
<td>UPT_HIGH_VOLTAGE_L3</td>
<td>0x0B</td>
<td>SHOULD be issued when high voltage L3 occurs</td>
</tr>
<tr>
<td>UPT_OVER_CURRENT_L1</td>
<td>0x0C</td>
<td>SHOULD be issued when over current L1 occurs</td>
</tr>
<tr>
<td>UPT_OVER_CURRENT_L2</td>
<td>0x0D</td>
<td>SHOULD be issued when over current L2 occurs</td>
</tr>
<tr>
<td>UPT_OVER_CURRENT_L3</td>
<td>0x0E</td>
<td>SHOULD be issued when over current L3 occurs</td>
</tr>
<tr>
<td>LogEvent Name</td>
<td>LogEvent Code</td>
<td>LogEvent Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>UPT_FREQUENCY_TOO_LOW_L1</td>
<td>0x0F</td>
<td>SHOULD be issued when frequency too low L1</td>
</tr>
<tr>
<td>UPT_FREQUENCY_TOO_HIGH_L1</td>
<td>0x10</td>
<td>SHOULD be issued when frequency too high L1</td>
</tr>
<tr>
<td>UPT_FREQUENCY_TOO_LOW_L2</td>
<td>0x11</td>
<td>SHOULD be issued when frequency too low L2</td>
</tr>
<tr>
<td>UPT_FREQUENCY_TOO_HIGH_L2</td>
<td>0x12</td>
<td>SHOULD be issued when frequency too high L2</td>
</tr>
<tr>
<td>UPT_FREQUENCY_TOO_LOW_L3</td>
<td>0x13</td>
<td>SHOULD be issued when frequency too low L3</td>
</tr>
<tr>
<td>UPT_FREQUENCY_TOO_HIGH_L3</td>
<td>0x14</td>
<td>SHOULD be issued when frequency too high L3</td>
</tr>
<tr>
<td>UPT_GROUND_FAULT</td>
<td>0x15</td>
<td>SHOULD be issued when ground fault occurs</td>
</tr>
<tr>
<td>UPT_BURST_DETECT</td>
<td>0x16</td>
<td>SHOULD be issued when burst detect alarm occurs</td>
</tr>
<tr>
<td>UPT_PRESSURE_TOO_LOW</td>
<td>0x17</td>
<td>SHOULD be issued when pressure too low</td>
</tr>
<tr>
<td>UPT_PRESSURE_TOO_HIGH</td>
<td>0x08</td>
<td>SHOULD be issued when pressure too high</td>
</tr>
<tr>
<td>UPT_FLOW_SENSOR_COMMUNICATION_ERROR</td>
<td>0x19</td>
<td>SHOULD be issued when flow sensor communication error occurs</td>
</tr>
<tr>
<td>UPT_FLOW_SENSOR_MEASUREMENT_FAULT</td>
<td>0x1A</td>
<td>SHOULD be issued when flow sensor measurement fault occurs</td>
</tr>
<tr>
<td>UPT_FLOW_SENSOR_REVERSE_FLOW</td>
<td>0x1B</td>
<td>SHOULD be issued when reverse flow is detected</td>
</tr>
<tr>
<td>UPT_FLOW_SENSOR_AIR_DETECT</td>
<td>0x1C</td>
<td>SHOULD be issued when flow sensor air detect alarm occurs</td>
</tr>
<tr>
<td>UPT_PIPE_EMPTY</td>
<td>0x1D</td>
<td>SHOULD be issued when pipe empty alarm occurs</td>
</tr>
<tr>
<td>UPT_INLET_TEMPERATURE_SENSOR_FAULT</td>
<td>0x1E</td>
<td>SHOULD be issued when inlet temperature sensor fault</td>
</tr>
<tr>
<td>UPT_OUTLET_TEMPERATURE_SENSOR_FAULT</td>
<td>0x1F</td>
<td>SHOULD be issued when outlet temperature sensor fault</td>
</tr>
</tbody>
</table>
12.4 **Pricing Function Set**

12.4.1 **Overview**

The pricing function set provides the tariff structures communicated by the server and is designed to support a variety of tariff types, including:

- Flat-rate pricing
- Time-of-Use tiers
- Consumption blocks
- Hourly day-ahead pricing
- Real-time pricing
- Combinations of the above

The Pricing Function Set supports application-specific tariffs for devices (e.g., PEV, DER), and special event-based prices like critical peak price (Note, as per [ZB 13-0201], CPP is treated as just another TOU tier).

The Pricing Function Set is designed to stand on its own but can be paired with the Metering, Billing and Prepayment function sets to provide additional benefit to users. The Pricing Function Set is not intended to provide all the information necessary to represent a premises's bill.

12.4.2 **List Ordering**

**Table 12-5 Pricing List Ordering.**

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>TariffProfile</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>RateComponent</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>TimeTariffInterval and</td>
<td>interval.start</td>
<td>creationTime</td>
<td>mRID</td>
</tr>
<tr>
<td>ActiveTimeTariffInterval</td>
<td>(ascending)</td>
<td>(descending)</td>
<td>(descending)</td>
</tr>
<tr>
<td>ConsumptionTariffInterval</td>
<td>startValue (ascending)</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

12.4.3 **Application Guidelines / Behavior**

12.4.3.1 **TariffProfile**

Pricing servers SHALL be capable of internally storing and supporting at least one TariffProfile instance.

Pricing clients SHALL be capable of internally storing and supporting at least one TariffProfile instance.

Pricing servers SHOULD be capable of internally storing and supporting at least three TariffProfile instances (e.g., multiple commodities, multiple service provider tariff options).
Pricing clients SHOULD be capable of internally storing and supporting at least three TariffProfile instances (e.g., multiple commodities, multiple service provider tariff options).

12.4.3.2 Rate Component

Pricing servers SHALL support at least two RateComponent instances for each TariffProfile.

Pricing clients SHALL support at least one RateComponent instance for each TariffProfile. Pricing clients supporting the use of this resource SHOULD support at least two instances of RateComponent (e.g., to convey prices for forward energy [consumed by premises] and reverse energy [supplied by premises]).

12.4.3.3 TimeTariffInterval

Rates that do not contain time-differentiated characteristics SHALL create one TimeTariffInterval instance with a DateTimeInterval of sufficient duration to cover at least the next 24 hours or use the maximum time value for duration to minimize data transmission.

Pricing servers SHALL support at least two TimeTariffInterval instances per RateComponent instance (e.g., the active and subsequent TimeTariffInterval instance for a RateComponent).

Pricing servers SHOULD support at least 48 TimeTariffInterval instances for at least one RateComponent instance.

Pricing servers SHALL provide at most one active TimeTariffInterval per rate component. TimeTariffIntervals are to be scheduled for each occurrence of a TOU. A given day would have a flow of TimeTariffIntervals for the TOU rates for that day. A particular TimeTariffInterval instance specifies the touTier that is in effect during that event's effective time period.

Pricing clients, upon detecting multiple active TimeTariffIntervals, SHALL ignore all but the TimeTariffInterval with the highest creation time. If this is insufficient to determine a unique active TimeTariffInterval (e.g., two active instances exist with the same creation time), clients SHALL operate as if there is no TimeTariffInterval defined for the given time period.

Pricing clients SHALL be capable of internally storing and supporting at least two TimeTariffInterval instances per RateComponent instance.

Pricing clients SHOULD be capable of internally storing and supporting at least five TimeTariffInterval instances per RateComponent instance.

The series of TimeTariffInterval instances on a server may contain gaps or breaks where pricing information is not defined for some time period. This may occur for various reasons, such as when private information is cleared from the server (e.g., during move-out) or potentially when superseding TimeTariffIntervals are created by the service provider (however, service providers should take care to ensure that gaps are not created, by creating additional TimeTariffInterval instances if necessary). The below guidelines promote common pricing client behavior and reduce the chances of different implementations displaying different cost information to a user in the event pricing information is unavailable during a particular period. This could cause users to question the reliability of the data. Rules for handling these gaps are as follows:

- If a pricing client displays the active or scheduled price or calculated instantaneous cost data to the user, the client SHALL indicate the presence of an unexpected issue with the price data.
- If a pricing client displays calculated running or averaged cost data to the user, the client MAY continue to display values by excluding the time periods where no price is defined. However, the
client SHALL also indicate the presence of an unexpected issue with the price data for as long as there are price gaps during the time period the client uses to calculate these values.

- Pricing clients SHALL NOT default to any hardcoded price attribute (e.g., $0) or use a price attribute from another (past / future) TimeTariffInterval for display or calculation. If a pricing client displays human-readable pricing information, then they SHALL display a non-numerical indicator (e.g., "XX", dashes, "NA").
- If the pricing server later makes TimeTariffInterval instances available to fill pricing gaps, pricing clients that display calculated cost information MAY recalculate past cost data based on the new information.

Pricing clients that are price responsive SHOULD return to normal operational mode (that is, the default behavior of the device without any price responsiveness) during time periods where no TimeTariffInterval instance is defined. These clients SHOULD provide some notification to the user that the active TimeTariffInterval instance price information is unknown.

The ActiveTimeTariffIntervalList only filters out inactive TimeTariffProfile instances; this SHALL NOT filter out ConsumptionTariffInterval instances. That is, if a client GETs a TimeTariffInterval from a server's ActiveTimeTariffIntervalList, the ConsumptionTariffIntervalList pointed to by that TimeTariffInterval's ConsumptionTariffIntervalListLink SHALL contain all ConsumptionTariffInterval instances (equal to the number of consumption blocks defined in the ReadingType), not just the one active ConsumptionTariffInterval instance.

12.4.3.4 Interval

While strongly RECOMMENDED, Pricing clients are NOT REQUIRED to follow the sign of randomization for Pricing function set messages. However, Pricing clients SHALL observe the absolute value of the randomizeDuration or randomizeStart value for the randomization range when calculating the randomization value. This allows more capable price clients to look ahead at scheduled prices (if available) and, using knowledge of the client's operating characteristics, determine if it is in the customer's best interest to react to the event earlier or later.

If, while a price-responsive client is acting upon a TimeTariffInterval, that TimeTariffInterval is cancelled, the client SHALL observe the randomizeDuration value when ceasing action.

12.4.3.5 ConsumptionTariffInterval

Pricing servers SHALL be capable of internally storing and supporting at least one ConsumptionTariffInterval element per TimeTariffInterval instance.

Pricing servers SHOULD be capable of internally storing and supporting at least five ConsumptionTariffInterval elements per TimeTariffInterval instance.

Pricing clients SHOULD be capable of internally storing and supporting at least five ConsumptionTariffInterval elements per TimeTariffInterval instance.

A particular TimeTariffInterval instances MAY NOT include a ConsumptionTariffIntervalListLink, meaning that ConsumptionTariffIntervals (and therefore the actual price) is not available for that TimeTariffInterval. In such a case, price responsive clients would be unable to act upon price; however, they MAY be price responsive to the touTier value (if present) in the TimeTariffInterval.

12.4.3.6 Sleepy Devices / Polling Clients

It is RECOMMENDED that sleepy pricing client devices send requests to the pricing server on a periodic basis. The RECOMMENDED time period for the periodic poll (for sleepy devices or other clients that do not or are unable to make use of subscriptions) is no more than once per hour but at least once per 24-
hour period. This ensures the client a high likelihood of receiving the pricing information needed to
manage its operations in a timely fashion while respecting limited network resources.

It is RECOMMENDED that polling pricing client devices request updated information for pending
TimeTariffInterval instances just prior to those TimeTariffInterval instances becoming active (e.g., 5-10
minutes prior, including any negative randomizeStart). This ensures the TimeTariffInterval instance
previously retrieved is still valid and accurate with the latest instance on the server.

12.4.3.7 Deployments with Multiple Pricing Servers

For the purposes of price responsiveness, clients SHOULD only follow one pricing server in the HAN per
commodity. Pricing clients MAY follow multiple Pricing servers for informational display purposes (e.g.,
to compare different providers) or price seeking behavior. More sophisticated devices (e.g., Premises
Energy Management System) MAY follow multiple Pricing servers and make policy based decisions to
dispatch local resources (e.g., Distributed Energy Resource) and / or provide a single Pricing server for
clients it controls based on user preferences.

Registered pricing devices SHALL determine their primary Pricing server via Function Set Assignments
and follow it for the purposes of price responsiveness. It is incumbent upon the user to choose the Pricing
server with which to register the device. Pricing devices SHALL periodically perform service discovery
to find new pricing servers with which it is registered and begin following them for the purposes of price
responsiveness. Pricing clients SHALL unsubscribe or discontinue following the previous Pricing server
for the purposes of price responsiveness. In addition to periodic discovery of new Pricing servers, it is
RECOMMENDED that devices allow a means of de-registration (or return to defaults) so the device can
be manually de-registered and not require the periodic polling time. If a client is not registered to any
Pricing server, the client SHALL use the Primacy value of any discovered public Pricing servers for the
commodity or commodities of interest.

When devices are registered to a Pricing server, they SHALL not act upon any "public" pricing servers
that are present in the HAN or become available.

12.4.3.8 Relative Pricing between Tiers and Blocks

Pricing servers using multiple TOU tiers SHALL associate higher prices with higher touTier values. That
is, the price of any (TOU Tier N, Consumption Block M) SHALL be less than or equal to the price of
(TOU Tier N+1, Consumption Block M). Note that this is only valid for comparing the same
consumptionBlock prices between two TOU tiers. Servers MAY be configured such that one or more
consumptionBlock prices from TOU tier N are greater than one or more consumptionBlock prices from
TOU tier N+1.

Similarly, there is no restriction regarding the relative prices between consumptionBlock prices within the
same TOU tier. That is, within one TOU tier, the price of consumptionBlock N MAY be greater than the
price of consumptionBlock N+1.

12.4.3.9 Price Responsiveness

Servers that also support EndDevice instances MAY include price response thresholds for a particular end
device. This provides a standard mechanism for end devices without user interfaces to receive
configuration data concerning customer preferences for price responsiveness. Servers MAY provide a
unique PriceResponseCfg resource for each RateComponent resource that server hosts.

Pricing clients that are registered to a particular Pricing program and are acting upon a particular
RateComponent SHOULD check their PriceResponseCfgList for a PriceResponseCfg resource
corresponding to the RateComponent. If a matching PriceResponseCfg is present, pricing clients
SHOULD consume the associated commodity when the price is less than the consumeThreshold value (typically used for scenarios such as negative pricing, pre-heating / cooling, and battery charging), and SHOULD reduce consumption to the maximum extent possible when the price is above the maxReductionThreshold value.

If an appropriate PriceResponseCfg is not present, or the present price is between the consumeThreshold and maxReductionThreshold values, pricing clients MAY base responsiveness on comparing the active tier value to the total number of tiers as specified in the ReadingType (as per the price-tier relationship defined in Section 12.4.3.8).

PriceResponseCfg servers SHALL NOT specify a consumeThreshold that is greater than or equal to the maxReductionThreshold. If a pricing client reads a PriceResponseCfg instance where the consumeThreshold is greater than or equal to the maxReductionThreshold, the client SHALL ignore the erroneous PriceResponseCfg instance.

12.4.4 LogEvents

Table 12-6 Pricing LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP_NO_TTI</td>
<td>0x00</td>
<td>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.</td>
</tr>
</tbody>
</table>

12.5 Messaging Function Set

12.5.1 Overview

This function set provides an interface for a text messaging service. Client devices of this function set include In-Premises Displays and other text messaging display devices. Server devices of this function set include ESIs or a back office server, depending on system design. The response function set is used in conjunction with the messaging function set to allow client devices to confirm the viewing of messages and report advanced responses.

Servers expose messages to client devices in the form of separate messaging URIs. Each messaging URI instance will contain information that a client device can use to display the message appropriately. For example, start time, duration of event, text to display, etc.

12.5.2 List Ordering

For list ordering purposes, the MessagingProgramList, TextMessageList and the ActiveTextMessageList SHALL be ordered based on the following criteria:

Table 12-7 Messaging List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessagingProgram</td>
<td>mRID (descending)</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>TextMessage and ActiveTextMessage</td>
<td>dateTimeInterval.start (ascending)</td>
<td>createdDateTime (descending)</td>
<td>mRID (descending)</td>
</tr>
</tbody>
</table>
12.5.3  **Application Guidelines / Behavior**

### 12.5.3.1 Messaging Program

MessagingProgram server and client devices SHALL be capable of internally storing and supporting at least 1 MessagingProgram instance. A MessagingProgram server and client device SHOULD be capable of supporting 3 unique MessagingProgram instances.

### 12.5.3.2 Text Message

MessagingProgram server devices SHALL be capable of internally storing and supporting at least 1 MessagingProgram instance. MessagingProgram server devices SHOULD be capable of internally storing and supporting 3 unique TextMessage instances. Additional information for common application features that the Messaging function set will perform can be found in the Common Application Functionality section of this document.

Messaging client devices SHALL be capable of internally storing and supporting at least 1 unique TextMessage instance, per supported MessagingProgram instance. As a highly RECOMMENDED recovery mechanism, when a maximum of storage of events has been reached and additional TextMessage instances are available on the server(s), clients SHOULD prioritize local storage and give preference to TextMessages with start times in the near future and to events with a higher PriorityType.

Devices can obtain text messages from many sources. It may be very common to have multiple Messaging functions set servers in a HAN. FunctionSetAssignments and MessagingProgram.primacy are used to indicate which messaging servers in the HAN the device should prioritize. MessagingPrograms indicated in FunctionSetAssignments SHALL be of higher priority than those found through discovery. MessagingProgram.priority SHALL be used as a secondary determinant of a messages priority.

### 12.5.4 LogEvents

There are no LogEvents generated by this function set.

### 12.6 Billing Function Set

#### 12.6.1 Overview

There are several resources that are used to support billing related functions. The billing function set provides consumption or costs, estimates of future consumption, and / or historical consumption from a service provider to an end device. In addition to consumption and costs that would be calculated by the back end systems and shared with the end devices, billing also provides a mechanism to allow the service provider to push down targets or challenges to help consumers manage their budgets. A target could be a percentage or fixed value of reduction – possibly chosen by the consumer - to meet within a defined time frame.

The TargetReading Resource provides a way for a service provider to create challenges or targets for an end customer to try to achieve within a certain time frame.

The ProjectionReading Resource provides a way for a service provider to provide future projected consumption or cost for a particular reading type that may be calculated outside of the HAN.

The HistoricalReading Resource provides a way for a service provider to provide historical consumption or cost for a particular reading type that is verified and possibly corrected at the service provider backend.

The CustomerAccount resource contains information specific to the account such as the currency. The Customer Account is associated with a list of Customer Agreements.
The Customer Agreement resource contains information about a particular agreement for service, at a particular Usage Point and a particular Tariff Profile rate.

A BillingPeriod relates to a timeframe that a commodity is billed on. There may be multiple BillingPeriods that relate to different TariffProfiles.

### 12.6.2 List Ordering

#### Table 12-8 Billing List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerAccount</td>
<td>customerName (ascending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>CustomerAgreement</td>
<td>serviceLocation (ascending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>BillingPeriod</td>
<td>interval.start (descending)</td>
<td>href (ascending)</td>
<td>N/A</td>
</tr>
<tr>
<td>HistoricalReading</td>
<td>description (ascending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>TargetReading</td>
<td>description (ascending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>ProjectionReading</td>
<td>description (ascending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>BillingReadingSet</td>
<td>timePeriod.start (descending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>BillingReading</td>
<td>timePeriod.start (ascending)</td>
<td>consumptionBlock (ascending)</td>
<td>touTier (ascending)</td>
</tr>
</tbody>
</table>

### 12.6.3 Application Guidelines / Behavior

#### 12.6.3.1 CustomerAccount and Customer Agreement Resources

A CustomerAccount is related to one customer (which may be an organization). Each CustomerAccount can have multiple CustomerAgreements associated, possibly representing different UsagePoints or commodities. Each CustomerAgreement can link the associated UsagePoint (Metering), Prepayment, TariffProfile (Pricing) and / or Billing information together. Servers SHALL support at least one CustomerAccount and one CustomerAgreement if the Billing function set is implemented, and SHOULD support at least three CustomerAgreements.

#### 12.6.3.2 BillingPeriod Resource

For each CustomerAgreement, there may be multiple BillingPeriods. Servers implementing the Billing function set SHALL support at least one BillingPeriod per CustomerAgreement, and SHOULD support at least three BillingPeriods per CustomerAgreement.

#### 12.6.3.3 TargetReading Resource

As a good practice, there should be only one TargetReading for a BillingPeriod. The values of the target readings will be an absolute measurement similar to a projected reading. If the service provider specifies the targets in a percentage reduction or other method it MUST be converted to an absolute value.
An example would be a customer who used 100kWh in a previous month with a reduction target of 10%.
This target would be specified in the TargetReading as 90kWh and it would be left to the device to
calculate the percentage or kWh reduction to which this equates.

12.6.3.4 ProjectedReading Resource
Projected Readings are tools for service providers to help a customer understand what their consumption
or cost might be projected into the future if all things within the home stayed fairly similar.
Examples of projected readings are:

- Consumption for the billing period.
- Cost of commodity for the billing period.
- On day X the consumption would be Y which would indicate a customer moving into a higher
  block tariff and thus a higher rate.
- The different TOU tiers at the end of the billing period based on current consumption habits.

These are just examples. Other projected readings could be created.

12.6.3.5 HistoricalReading Resource
Historical Readings are meant to provide a resource so that a service provider can provide consumption or
cost that has occurred in the past and could be validated and corrected by backoffice systems. Examples
of this would be:

- Previous day
- Previous month
- Previous billing period
- Previous year
- TOU A, TOU B, etc.
- Block 1, Block 2, etc.
- Cost of consumption for TOU A for the billing period

This is just a sample of what could be used. Because the information will be provided from the backend
system, it can be of billing quality or near billing quality. End devices that read consumption from
metering end points will be allowed to change their local consumption reading to match information from
the billing resource to allow for closer to actual billed consumption. Servers implementing the Billing
function set SHALL support at least one HistoricalReading and one associated ReadingType,
BillingReadingSet, and BillingReading with at least one Charge, and SHOULD support at least three of
each of these, multiplied by parent containers where appropriate (e.g., three HistoricalReadings with three
BillingReadingSets per HistoricalReading for a total of nine BillingReadingSets, etc.)
12.6.3.6 **Deployments with Multiple Billing Servers**

EndDevices that discover multiple Billing servers SHOULD differentiate between those sources on user displays of the information, unless the objects have the same mRID.

12.6.4 **LogEvents**

There are no LogEvents generated by this function set.

12.7 **Prepayment Function Set**

12.7.1 **Overview**

The Prepayment function set defines a mechanism for the conditional delivery of services based upon outstanding credit or debt. It provides an interface for appropriately privileged clients to view, update or act upon account balance information. Accounting in prepayment systems may be measured on a monetary basis (e.g., Dollars or Euros remaining) or on a commodity basis (e.g., kilowatt-hours or BTUs remaining). A service-providing device (typically a meter) can use the account balance information from a prepayment server in combination with consumption and price data to determine if service should continue. In some scenarios ("Local" mode), the service-providing device and the prepayment server are the same. Alternatively, the continuation of service may be directed by an external prepayment server, either in response to local calculation ("ESI" mode) or to an out-of-band signal from the service provider ("Central Wallet" mode). Client devices that provide a user interface to the service provider's customers may use the prepayment function set to display information about remaining balances or to request additional credit (in some scenarios, through the transmission of payment tokens; however, note that SEP 2 only provides a wrapper for this token data, the mechanism by which tokens are produced and consumed is out of scope for this specification).

12.7.2 **List Ordering**

Table 12-9 Prepayment List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepayment</td>
<td>mRID (descending)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Credit Register</td>
<td>effectiveTime (descending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>SupplyInterruptionOverride</td>
<td>interval.start (ascending)</td>
<td>interval.duration (ascending)</td>
<td>N/A</td>
</tr>
</tbody>
</table>

12.7.3 **Application Guidelines / Behavior**

12.7.3.1 **General**

Servers implementing the Prepayment function set SHALL be capable of internally storing and supporting at least one Prepayment instance. Prepayment servers SHALL support one and only one AccountBalance resource per Prepayment instance.

A server implementing the Prepayment function set SHALL support one and only one PrepayOperationStatus resource per Prepayment instance.

Servers implementing the Prepayment function set in Local or ESI mode SHALL be capable of internally storing and supporting at least one CreditRegister instance per Prepayment instance.
Prepayment clients MAY POST to the CreditRegisterList to transmit a new CreditRegister transaction.

A Prepayment client MAY PUT to PrepayOperationStatus to switch between using regular credit and emergency credit. Typically, this interface is used by service customers to tap a backup credit pool when normal credit is exhausted and cannot be immediately recharged. Prepayment server implementers MAY apply additional vendor-specific rules around when this mode of operation may be changed (e.g., emergency credit might only be allowed when availableCredit is less than or equal to the creditExpiryLevel).

12.7.3.2 **Prepayment Server / Usage Point Server Communication**

In most Prepayment configurations, client behavior is minimally affected by server state. That is, the typical client is an IHD that displays the present account and status data, posts new credit transactions, requests the use of emergency credit, or displays historical transaction data. However, in the ESI prepayment mode (and in some configurations of the Central Wallet mode), external meters (or other service-providing devices), which may be Usage Point servers, are also prepayment clients. These clients are significantly affected by server state. These devices SHALL monitor the server’s Operation Status resource and should react appropriately to changes of the serviceStatus and serviceChange elements. This reaction includes connecting or disconnecting service.

12.7.3.3 **Mirroring Behavior**

In most Smart Energy 2.0 function sets, mirroring involves behavior similar to the following pattern:

1) Device B issues an OPTIONS method to determine if Server A supports a POST to a given resource list.

2) Device B posts its data to the resource list on Server A. Server A creates a mirrored resource for Device B.

3) Client C reads Device B data from the mirrored resource on Server A.

The Prepayment function set, in some deployments, requires additional mirroring behavior. With Prepayment, mirror instances MAY need to act as mailboxes for the mirrored server, such as in the following scenario:

1) Device B issues an OPTIONS method to determine if Server A supports a POST to a given resource list.

2) Device B posts its data to the resource list on Server A. Server A creates a mirrored resource for Device B.

3) Client C POSTS data to the mirrored resource on Server A.

4) Device B reads the mirrored resource on Server A to obtain Client C data.

One use case for this mode is when a sleepy meter is implementing a Prepayment server in local prepayment mode. This means that the meter will be expecting CreditRegister transactions to be posted by prepayment clients. However, sleepy devices are unable to receive transmissions while idle, and a sleepy server will typically be interacting with other HAN devices through a mirrored instance. In this scenario, prepayment clients SHALL post the CreditRegister transactions to the mirrored instance. The server hosting the mirrored instance SHOULD maintain the instances of CreditRegister transactions for at least 72 hours, so that the sleepy meter MAY download and act upon the credit transactions. Note that for
all intents and purposes (including discovery) that to Prepayment clients on the HAN, the mirrored instance is the prepayment server; the other clients are likely unaware of the sleepy device.

12.7.4 **LogEvents**

Table 12-10 Prepayment LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPY_CREDIT_POST_SUCCESS</td>
<td>0x00</td>
<td>SHOULD be issued when a client successfully posts a CreditRegister to the CreditRegisterList.</td>
</tr>
<tr>
<td>PPY_CREDIT_POST_FAIL</td>
<td>0x01</td>
<td>SHOULD be issued when a client posts an invalid resource to the CreditRegisterList.</td>
</tr>
</tbody>
</table>

12.8 **Energy Flow Reservation Function Set**

12.8.1 **Overview**

This function set provides an interface for exchange of energy flow (e.g., charge or discharge) reservation events. Client devices of this function set include Plug-in Electric Vehicles, Distributed Energy Storage devices, and other managed loads that draw large amounts of power. Server devices of this function set include ESIs, EVSEs, and EMSs. FlowReservations allow for the scheduling of high demand periods such as during fast-charging transactions, to make them run at different times and avoid high aggregated demand. Typically, energy rates have penalties, charges, or customer classes for different demand tiers, so it is usually least expensive to keep the maximum demand as low as possible. Distribution utilities may support this function set to minimize the maximum demand across the distribution system.

Servers accept FlowReservationRequests from client devices by exposing a FlowReservationRequestList for each EndDevice. Clients POST a request to transfer a certain amount of energy during a specific interval, at a specific rate. Servers create an associated FlowReservationResponse in the EndDevice's FlowReservationResponseList. Servers may create superseding events to modify the interval within the requested timeframe and update the status to affect client behavior and distribute load across multiple reservations. To do this, the server must have knowledge of multiple clients, but can simply approve all requests unchanged if there is no other information.

12.8.2 **List Ordering**

Table 12-11 FlowReservation List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowReservationRequest</td>
<td>interval.start</td>
<td>creationTime</td>
<td>mRID</td>
</tr>
<tr>
<td></td>
<td>(ascending)</td>
<td>(descending)</td>
<td>(descending)</td>
</tr>
<tr>
<td>FlowReservationResponse</td>
<td>interval.start</td>
<td>creationTime</td>
<td>mRID</td>
</tr>
<tr>
<td></td>
<td>(ascending)</td>
<td>(descending)</td>
<td>(descending)</td>
</tr>
</tbody>
</table>

12.8.3 **Application Guidelines / Behavior**

12.8.3.1 **FlowReservationRequest**

A client generates a FlowReservationRequest in order to trigger a FlowReservationResponse event from the server.

FlowReservation server and client devices SHALL be capable of internally storing and supporting at least one FlowReservationRequest instance and one FlowReservationResponse instance. A FlowReservation
server and client devices SHOULD be capable of internally storing and supporting at least three unique FlowReservationRequest instances.

Clients SHALL NOT modify a FlowReservationRequest except to update the associated RequestStatus. Clients SHALL update the associated RequestStatus to Cancelled for any FlowReservationRequest that they want a server to subsequently disregard. A server SHALL return a 400 (“Bad Request”) response code if it receives a PUT of a FlowReservationRequest that contains changes other than RequestStatus.

If a FlowReservationRequest is removed from the server, clients and servers SHALL NOT assume the FlowReservationRequest has been cancelled. Servers MAY remove a request as required (e.g., after the request has been fulfilled or space is needed). It is the server’s responsibility to manage FlowReservationRequests. Servers SHOULD remove FlowReservationRequests when the associated FlowReservationResponse expires.

12.8.3.2 FlowReservationResponse

FlowReservation server and client devices SHALL be capable of internally storing and supporting at least one FlowReservationResponse instance. FlowReservation server and client devices SHOULD be capable of internally storing and supporting at least three unique FlowReservationResponse instances.

A FlowReservationResponse SHALL be created in response to each FlowReservationRequest. If a server wants to deny a request, it SHALL create a FlowReservationResponse with duration equal to zero.

12.8.4 LogEvents

Table 12-12 Flow Reservation LogEvents.

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_SCHEDULING_ERROR</td>
<td>0x00</td>
<td>SHOULD be issued if the server encounters an error and is unable to schedule reservations normally.</td>
</tr>
</tbody>
</table>

12.9 Distributed Energy Resources Function Set

12.9.1 Overview

This function set provides an interface to manage Distributed Energy Resources (DER). There are two main types of client devices of this function set: generation and storage. Examples of the first type include fuel cells, intelligent solar inverters, and backup generation units. Examples of the second type include battery storage systems and electric vehicles (which may not be capable of discharging). Server devices of this function set include ESIs and premises energy management systems.

Servers host one or more DERPrograms, which in turn expose DERControl events to DER clients. DERControl instances contain attributes that allow DER clients to respond to events that are targeted to their device type. A DERControl instance also includes scheduling attributes that allow DER clients to store and process future events. These attributes include start time and duration, as well an indication of the need for randomization of the start and / or duration of the event.

The SEP 2 DER client model is based on the SunSpec Alliance Inverter Control Model [SunSpec] which is derived from IEC 61850-90-7 [61850] and [EPRI]. As these specifications are under development at the present time, it is likely that differences will exist between the published standards. The reader is
referred to [EPRI], [61850], and [SunSpec] for a detailed description of the DER features referred to in this function set.

12.9.2  Terminology and Conventions

The terminology used to describe the SEP 2 DERControl interface is relative to the DER as a power producer. A DER described here as a generator delivers active AC power for consumption in the residence or the grid. By convention, a sub-meter connected at the DER accumulates positive energy usage when the DER is delivering active power. From the utility perspective a DER operating in this mode may be viewed as a "negative load" and the premises aggregation meter will accumulate energy usage at a slowed or negative rate.

When the DER has attached storage, it is described here as a load and receives active power when in charging mode. It behaves like a generator and delivers active power when in discharging mode.

By convention, positive active and reactive powers flow in the same direction (here the reference direction is from the DER to the utility). When a DER is providing positive vars (i.e. behaving like an over-excited motor or generator) it is said here to be delivering reactive power (VAR).

In addition to the reference frame, the SEP 2 DERControls interface defines the Power Factor (PF) sign convention used between servers and clients in order to avoid configuration mismatches. DERControl instances that affect PF assume the EEI [61850] sign convention. Negative ("lagging") PF indicates that both active and reactive powers are flowing in the same direction (either both delivered or received). Note that the EEI sign convention treats unity PF as unsigned. It may be necessary for the DER client to locally translate PF sign before issuing commands to an associated target device.

These relationships are shown in Figure 12-1. At a given point in time, a DER may operate in any one of 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

#### DERProgram List Ordering

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>DERProgram</td>
<td>primacy (ascending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>DERControl and ActiveDERControl</td>
<td>interval.start (ascending)</td>
<td>creationTime (descending)</td>
<td>mRID (descending)</td>
</tr>
<tr>
<td>DERCurve</td>
<td>creationTime (descending)</td>
<td>mRID (descending)</td>
<td>N/A</td>
</tr>
<tr>
<td>DER</td>
<td>href (ascending)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### 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.
DERProgram server devices SHALL be capable of internally storing and supporting at least one DERProgram instance. DERProgram server devices SHOULD be capable of internally storing and supporting three unique DERProgram instances. Each DERProgram instance SHALL be uniquely identified by an mRID.

DER client devices SHALL be capable of internally storing and supporting at least one DERProgram instance. DER client devices SHOULD be capable of internally storing and supporting two unique DERProgram instances.

**12.9.4.2 DERControl**

A DERProgram exposes control parameters to a DER client via DERControl Events. At any point in time a DER client is managed by a single DERControl Event, which SHALL supersede any previous Event. A DER client MAY reject or partially act upon a DERControl Event based on its capabilities. The control mode(s) supported by a DER may be determined from its DERCapability.modesSupported attribute. If there are no active Events, the DER client SHALL be managed by the DefaultDERControl instance exposed by the preferred DERProgram.

DERProgram server devices SHALL be capable of internally storing and supporting at least five unique DERControl instances, which MAY be distributed among multiple programs. DERProgram server devices SHOULD be capable of internally storing and supporting a total of 10 unique DERControl instances. Each DERProgram SHALL internally store a single DefaultDERControl instance that defines the default behavior of associated DER clients when no active Events are available. Each DERControl and DefaultDERControl instance SHALL be uniquely identified by an mRID.

DER client devices SHALL be capable of internally storing and supporting at least one DERControl instance and a single DefaultDERControl instance for each stored DERProgram. DER client devices SHOULD be capable of internally storing and supporting three DERControl instances. DER clients SHOULD prioritize local storage and give preference to DERControls with start times in the near future.

DERControl modes are divided into two categories. Immediate control modes (opModFixedW, opModFixedPF, opModFixedVar, opModFixedFlow) request a fixed output setting that the DER client SHOULD attempt to satisfy. Curve-based control modes allow a DER client to dynamically vary an output (dependent variable) as a function of a given input signal (independent variable). For example, an intelligent inverter may support the Volt-Watt mode and dynamically control its active power delivery based on local voltage measurements. Curve-based control modes are based on DERCurve instances.

**12.9.4.3 DERCurve**

The DERCurves associated with a given DERProgram are grouped under the program's DERCurveList resource. Each DERCurve SHALL contain a defined curveType value that associates the curve with a given control mode and implicitly defines the units of measure that apply to its curveData points.

A DERCurve SHALL specify an array of one or more curveData points. Watt-PowerFactor curve types SHALL specify two dependent variables (yvalue plus excitation) per curveData point. All other curve types define a single independent variable (xvalue) and dependent variable (yvalue) per curveData point. For example, an implementation of the Volt-Watt mode and dynamically control its active power delivery based on local voltage measurements. Curve-based control modes are based on DERCurve instances.

Implementations SHALL support a minimum of four curves having a minimum of 10 points per curve for each curve-based DERControl mode supported, unless otherwise specified.

As shown in Figure 12-2, an array of points may be used to represent a piecewise linear curve with hysteresis. This allows flexibility in defining stable behavior, differences in ramp rates, etc.
Figure 12-2: Example Volt-VAr Curve and Hysteresis

Figure 12-3 shows the DER operating regions defined by LVRT / HVRT curves. LVRT / HVRT curves are assumed to extend horizontally to the left to zero seconds before the first point in the array and to the right horizontally after the right-most point in the array.

Figure 12-3: Example Low and High - Voltage Ride Through Curves
12.9.4.4 **DER Info Resources**

A DER MAY be modeled as one or more DER client instances. For example, a device that consists of a solar inverter with attached storage may be modeled as separate or combined generation and storage DERs, depending on the complexity of the device's local control interface. The DER resource exposes the capability limits of a specific Distributed Energy Resource, as well as basic settings, status, and availability. A DER client SHOULD store these resources locally.

The currently executing DERProgram SHALL be referenced by the DER's CurrentDERProgramLink.

This DERProgram instance SHOULD reference local copies of the active DERControl and DERCurves.

12.9.4.4.1 **DERCapability**

The DER resource exposes the capabilities of a specific Distributed Energy Resource, referred to as its "nameplate ratings". Ratings are read-only values established by the DER manufacturer by design or manufactured configuration, e.g. the continuous delivered active power rating capability in watts (rtgW), and are available by reading the DERCapability resource.

12.9.4.4.2 **DERSettings**

The DERSettings resource provides a means to adjust the operating limits of a DER device as established by its nameplate ratings. For example, the active power output (setMaxW) may be reduced or increased as a function of attached photo-voltaic panels, condition of the equipment, season of the year, or intended use, subject to the maximum limit by rtgMaxW.

The basic settings also include configuration settings related to a specific installation such as setVRef, the nominal voltage at the point of common coupling; setVRefOfs, the voltage difference from the point of common coupling to the electrical connection point of the inverter; and the set point for the nominal frequency. Each rating value in a DER's DERCapability instance MAY have a corresponding setting value in its DERSettings instance (which equals the rating value by default). A modified rating SHALL have a corresponding setting.

12.9.4.4.3 **DERStatus**

The DER resource references a DERStatus instance that contains basic operational status attributes for the DER device. Information such as accumulated generation readings SHOULD be made available by a sub-meter referenced by the DER's AssociatedUsagePointLink.

12.9.4.4.4 **DER Availability**

The DERAvailability object is used by client devices to report their availability to deliver reserve active and reactive power. It MAY also be exposed instead by devices that are able to report this information on behalf of other devices. Duration attributes MAY be provided to indicate how long the generation can be sustained.

12.9.5 **DER Client Device Requirements**

DER device architectures are expected to vary widely. Therefore, minimum required functionality is based on DER type. A DER instance SHOULD be as simple as possible, but no simpler. For example, if a generator type DER can deliver reactive power and supports fixed VAr control mode then it must include rtgVAr and setMaxVAr.

If there is a significant difference between delivered and received VAr capability, then the DER MAY also include rtgVArNeg and setMaxVArNeg or use rtgVAr to specify a common (minimum) rating for both delivered and received VAr. Similarly, if a device includes attached storage and the charging and discharging mode rating limits differ significantly then the device can be modeled as separate DERs.
Each unique DER instance SHALL link to a DERCapability instance that SHALL contain type, modesSupported, rtgA, and rtgW attributes. The type attribute determines which additional modes and attributes are required as shown in the tables below.

The tables are interpreted as follows. If the optionality key in the column Opt1 is "M" (mandatory) then the DER SHALL support the control mode(s) listed under modesSupported; if the key is "R" (recommended) then the DER SHOULD support the listed control mode; if the key is "O" (optional) then the DER MAY support the listed control mode.

If the optionality key in the column Opt2 is "M" (mandatory) then the DER SHALL include the attribute(s) listed in the column Related attribute(s); if the key is "R" (recommended) then the DER SHOULD support the listed attribute; if the key is "O" (optional) then the DER MAY support the listed attribute.

If multiple modes are listed in a given entry, then support for any one of them meets the requirement.

Each mode will then be individually listed together with its required attributes. If multiple attributes are listed in a given entry, then support for any one of them meets the requirement. For example, a storage DER that supports only charging (e.g. a PEV) may choose to omit rtgMaxChargeRate and expose its maximum charging rate via the rtgW attribute. However, if a storage DER also supports discharge mode then it MUST use rtgMaxChargeRate to expose its charging rate and MAY use either rtgMaxDischargeRate (if the DER supports simultaneous generation and discharge) or rtgW to expose its maximum discharging rate.

Table 12-14 Modes and Attributes for Generator Type DERs.

<table>
<thead>
<tr>
<th>Opt1</th>
<th>modesSupported</th>
<th>Opt2</th>
<th>Related attribute(s)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>12 - Fixed W</td>
<td>M</td>
<td>rtgW/setMaxW</td>
<td>Continuous active power output (includes max discharge rate if combined generator / storage type)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O</td>
<td>rtgVA/setMaxVA</td>
<td>Continuous apparent power output</td>
</tr>
<tr>
<td>M</td>
<td>3 - Volt-Watt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>13 - Fixed VAr or 14 - Fixed PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13 - Fixed VAr</td>
<td>M</td>
<td>rtgVAr/setMaxVAr</td>
<td>If Fixed VAr mode is supported, then Volt-VAr mode SHOULD be supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O</td>
<td>rtgVArNeg/setMaxVArNeg</td>
<td>SHOULD be included if received and delivered VAr differ significantly</td>
</tr>
<tr>
<td>14 - Fixed PF</td>
<td>M</td>
<td>rtgMinPF/setMinPF</td>
<td>If Fixed PF mode is supported, then Watt-PF mode SHOULD be supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>O</td>
<td>rtgMinPFNeg/setMinPFNeg</td>
<td>SHOULD be included if received and delivered VAr differ significantly</td>
</tr>
</tbody>
</table>

Page 128 Copyright © 2013, ZigBee Alliance, Inc. and HomePlug Powerline Alliance, Inc. All rights reserved.
Most DERs will typically act as generators, with the possible exception of DERType 81 (electric vehicle).

### Table 12-15 Modes and Attributes for Storage Type DERs.

<table>
<thead>
<tr>
<th>Opt1</th>
<th>modesSupported</th>
<th>Opt2</th>
<th>Related attribute(s)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>15 - Charge mode</td>
<td>M</td>
<td>rtgMaxChargeRate/setMaxChargeRate or rtgW/setMaxW</td>
<td>rtgMaxChargeRate is &quot;M&quot; if Discharge mode is supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M</td>
<td>rtgWh or rtgAh</td>
<td>Storage capacity</td>
</tr>
<tr>
<td>O</td>
<td>16 - Discharge mode</td>
<td>M</td>
<td>rtgMaxChargeRate/setMaxChargeRate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>M</td>
<td>rtgMaxDischargeRate/setMaxDischargeRate or rtgW/setMaxW</td>
<td>rtgMaxDischargeRate is &quot;M&quot; if combined generator / storage type</td>
</tr>
</tbody>
</table>

Storage type DERs include DERType 80 (immobile storage), 81, and 82 (combined PV and storage).

Note that remote connect and disconnect functions for generator and storage DERs are not control modes but settings, and SHOULD be included in DERSettings.

#### 12.9.6 LogEvents

There are no LogEvents generated by this function set.

### 12.10 Metering Mirror

#### 12.10.1 Overview

The Metering Mirror function set provides a mechanism for constrained devices to post metering data to a Metering server in a very efficient manner. Great effort has gone into minimizing the number of transactions needed to create and maintain the mirroring relationship. Therefore mechanisms and structures of this function set differ from other function sets to attain this efficiency.

#### 12.10.2 List Ordering

#### Table 12-16 Metering Mirror List Ordering.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Primary Key</th>
<th>Secondary Key</th>
<th>Tertiary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>MirrorUsagePoint</td>
<td>mRID</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>(descending)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 12.10.3 Application Guidelines / Behavior

A Metering Mirror function set server SHALL NOT advertise support for mirroring unless it has the resources available to host at least one additional mirror. The server must have room for at least one instance of each of the resources possible under a Usage Point.

The following rules apply to creating and maintaining Metering Mirrors.

1. To create a new Metering Mirror the client SHALL POST to the server's MirrorUsagePointList (e.g., /mup) for the mirrored usage point.
a) This POST SHALL contain at least the information through the definition of
MirrorMeterReadings and ReadingType including the MirrorUsagePoint mRID and
MirrorMeterReading mRIDs.

b) The POST MAY also contain MirrorReadingSets and Readings.

c) If the mRID of the MirrorUsagePoint is unique (does not match a
MirrorUsagePoint.mRID of an existing MirrorUsagePoint) the response SHALL be
response code 201 (Created), the MirrorUsagePoint URI SHALL be included in the
Location header.

d) If the mRID of the MirrorUsagePoint matches an existing MirrorUsagePoint, the new
data SHALL be written over the existing MirrorUsagePoint (and associated UsagePoint)
and the response code SHALL be 204 (No Content), the MirrorUsagePoint URI SHALL
be included in the Location header. If the MirrorUsagePoint contains
MirrorMeterReadings, then the guidance of rules 8 and 9 are to be applied.

2) When the Metering Mirror function set server receives a POST it SHALL copy the received data,
including mRIDs, into the normal metering structure to its Metering UsagePoint structure
(e.g., /upt), and it SHALL allocate enough resources to manage the mirror and its data.

3) A GET of the resource (MirrorUsagePoint) identified in the response to the initial POST SHALL
return a resource with only the first level elements (i.e., sub-elements and collections are not
included).

4) To POST new data to an existing MirrorUsagePoint, the Metering client SHALL POST a
MirrorMeterReading or MirrorMeterReadingList containing MirrorReadingSets and/or Readings
to the resource identified in the Metering server's response to the POST that created the resource
(e.g., /mup/3).

5) The Metering Mirror server SHOULD only accept POSTs to a given MirrorUsagePoint from the
client that created the mirror.

6) If a POST to the MirrorUsagePoint is of a MirrorMeterReading then a successful response
SHALL contain a Location header indicating the URI of the MeterReading resource under the
associated UsagePoint (e.g., /upt/2/mr/3).

7) If a POST to the MirrorUsagePoint is of a MirrorMeterReadingList then a successful response
SHALL contain a Location header indicating the URI of the MeterReadingList under the
associated UsagePoint (e.g., /upt/2/mr).

8) In a POST to the MirrorUsagePoint, the mRID attribute of the MirrorMeterReading(s) SHALL be
used by the Metering Mirror server to associate the data in a POST with the MeterReading in the
associated UsagePoint.

   a) In a POST to the MirrorUsagePoint, if the mRID attribute matches a previous
MirrorMeterReading then the contained MirrorReadingSets SHALL be added to the
associated MeterReading and a contained Reading SHALL replace the existing Reading.
The contents of the MirrorMeterReading shall overwrite the data in the associated
MeterReading.

   b) In a POST to the MirrorUsagePoint, if the mRID does not match a previous
MirrorMeterReading and it contains a ReadingType, a new MeterReading SHALL be
created under the associated UsagePoint with the new data.
c) In a POST to the MirrorUsagePoint, if the mRID does not match a previous MirrorMeterReading and there is not a ReadingType then the request SHALL be rejected with a response code 400 (Bad Request).

9) In a POST to the MirrorUsagePoint, where the request is not rejected, the new data SHALL be applied to the related UsagePoint resource structure according to the following:

a) If a MirrorReadingSet is received with a duplicate mRID of an existing ReadingSet, and it is targeted within the same resource hierarchy, then the new data SHALL replace the existing data of the identified ReadingSet.

b) If a MirrorReadingSet is received with a unique mRID then the new data SHALL be added to the identified ReadingSetList.

10) If a client POSTs more data than the Metering Mirror server is willing to accept, the server SHALL respond with a response code of 413 (Request Entity Too Large).

11) The Metering Mirror server MAY decide when to remove data from the related UsagePoint resource structure.

A Metering Mirror function set server MAY implement a timeout mechanism on a mirror. If a Metering Mirror function set server does not receive any POSTs from a Metering Mirror function set client for more than a specified time the server MAY remove the MirrorUsagePoint resource and its related UsagePoint resource. The recommended timeout is 72 hours.

12.10.4 **LogEvents**

**Table 12-17 Metering Mirror LogEvents.**

<table>
<thead>
<tr>
<th>LogEvent Name</th>
<th>LogEvent Code</th>
<th>LogEvent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUP_MIRROR_EXPIRED</td>
<td>0x00</td>
<td>SHOULD be generated by a server when a Metering Mirror expires.</td>
</tr>
</tbody>
</table>
13 Manufacturer – Specific Proprietary Extensions

13.1 Overview
This section describes rules and mechanisms for interested parties to extend the Smart Energy Profile 2.0 with proprietary extensions.

It should be noted that as the Smart Energy Profile 2.0 is intended to run over an IP stack, many techniques already exist for providing proprietary services over such a stack with various protocols. This section is intended for guidance to developers of proprietary extensions that may wish to be similar to the design of the Smart Energy Profile 2.0 or add extended elements to the Smart Energy Profile 2.0.

13.2 xDNS/DNS-SD
Proprietary extensions SHALL NOT be made using the "smartenergy" Service Type.

It is RECOMMENDED that proprietary extensions that wish to use xDNS/DNS-SD apply for a new Service Type with which to operate.

13.3 URIs
As URIs are dynamically discovered and used, proprietary extensions are free to place proprietary resources at URIs of their choosing. It is RECOMMENDED that proprietary resources not be placed at URIs 'RECOMMENDED' in this specification and in [ZB 13-0201].

13.4 Resources
Proprietary extensions SHALL NOT place any objects, elements, attributes, etc. in the standardized SEP XML namespace ("http://zigbee.org/sep") and instead SHALL be placed in a different XML namespace.

The following examples demonstrate allowed and disallowed extensions. In these examples, "SEPElement(#)") is used to demonstrate elements that are defined in the SEP schema.

"MFEElement(#)") and "MFENS" are used to demonstrate elements and namespaces that are proprietary extensions respectively.

The example given below demonstrates allowed and disallowed (crossed out) element extensions.

```
<SEPElement1 xmlns="http://zigbee.org/sep"
  xmlns:MFENS="http://foo.org/mfe">
  <SEPElement2>a</SEPElement2>
  <SEPElement3>b</SEPElement3>
  <MFENS:MFEElement1>c</MFENS:MFEElement1>
</SEPElement1>
```

```
<SEPElement1 xmlns="http://zigbee.org/sep">
  <SEPElement2>a</SEPElement2>
  <SEPElement3>b</SEPElement3>
  <MFEElement1>c</MFEElement1>
</SEPElement1>
```

Figure 13-1: Allowed and Disallowed Extension

Proprietary extensions SHALL NOT extend enumerations defined in the SEP 2 schema.
Proprietary extensions made to standardized objects (in a proprietary namespace) defined in the SEP 2 schema SHALL be able to be ignored. That is, a device that does not understand a proprietary extension can safely ignore the extension.

Proprietary extensions made to standardized objects (in a proprietary namespace) defined in the SEP 2 schema SHALL NOT change the semantics of elements and attributes defined in the SEP 2 schema.

13.5 **DeviceCapabilities Resource**

It is RECOMMENDED that proprietary extensions use a different resource in which to list further resources.

Should a proprietary extension wish to use the standard DeviceCapabilities resource, it MUST do so following the same rules as for other resources.
14 Appendix A - Web-Application Description Language (INFORMATIVE)

Note that the WADL (sep_wadl.xml, contained in [ZB 13-0201]) is NORMATIVE. This section presents a human-friendly view of the information to facilitate understanding.

14.1 Support Resources Section

14.1.1 Device Capability Function Set

14.1.1.1 DeviceCapability Resource

Used to determine the resources available on a server.

Sample URI: /dcap

Request Representation: DeviceCapability
Response Representation: DeviceCapability
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.1.2 Self Device Resource Function Set

14.1.2.1 SelfDevice Resource

The device that is providing the services being accessed.

Sample URI: /sdev

Request Representation: SelfDevice
Response Representation: SelfDevice
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

14.1.3 End Device Resource Function Set

14.1.3.1 EndDeviceList Resource

End device resource list.

Sample URI: /edev

Request Representation: EndDevice
Response Representation: EndDeviceList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.1.3.2 EndDevice Resource

End device instance.

Sample URI: /edev/{id1}

Request Representation: EndDevice
Response Representation: EndDevice
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

14.1.3.3 Registration Resource

Contains registrations related to the indicated device.

Sample URI: /edev/{id1}/rg

Request Representation: Registration
Response Representation: Registration
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional
14.1.3.4 **DeviceStatus Resource**

Contains the current operational state of the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/dstat
Request Representation: DeviceStatus
Response Representation: DeviceStatus
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

14.1.4 **Function Set Assignments Function Set**

14.1.4.1 **FunctionSetAssignmentsList Resource**

Contains function set assignments present on the server and/or related to the indicated device.

Sample URI: /edev/{id1}/fsa
Request Representation: FunctionSetAssignments
Response Representation: FunctionSetAssignmentsList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.1.4.2 **FunctionSetAssignments Resource**

Contains links to the specific function set assignments.

Sample URI: /edev/{id1}/fsa/{id2}
Request Representation: FunctionSetAssignments
Response Representation: FunctionSetAssignments
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.1.5 **Subscription/Notification Mechanism Function Set**

14.1.5.1 **SubscriptionList Resource**

Contains subscriptions related to the indicated device. Documented in Subscription / Notification Mechanism.

Sample URI: /edev/{id1}/sub
Request Representation: Subscription
Response Representation: SubscriptionList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

14.1.5.2 **Subscription Resource**

A specific subscription

Sample URI: /edev/{id1}/sub/{id2}
Request Representation: Subscription
Response Representation: Subscription
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory

14.1.5.3 **NotificationList Resource**

A list of notifications

Sample URI: /ntfy
Request Representation: Notification
Response Representation: NotificationList
Methods: GET/HEAD: Discouraged, PUT: Error, POST: Mandatory, DELETE: Error
14.1.5.4 Notification Resource

A specific notification

Sample URI: /ntfy/{id1}

Request Representation: Notification
Response Representation: Notification

Methods: GET/HEAD: Discouraged, PUT: Error, POST: Error, DELETE: Error

14.1.6 Response Function Set

14.1.6.1 ResponseSetList Resource

List of ResponseSet instances or channels. Devices implementing the ResponseSetList resource MAY support multiple instances of ResponseSet

Sample URI: /rsps

Request Representation: ResponseSet
Response Representation: ResponseSetList

Methods: GET/HEAD: Optional, PUT: Error, POST: Discouraged, DELETE: Error

14.1.6.2 ResponseSet Resource

Specific ResponseSet instance. This resource can be thought of as a particular ResponseList or channel.

Sample URI: /rsps/{id1}

Request Representation: ResponseSet
Response Representation: ResponseSet

Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Discouraged

14.1.6.3 ResponseList Resource

List of Response instances.

Sample URI: /rsps/{id1}/rsp

Request Representation: Response
Response Representation: ResponseList

Methods: GET/HEAD: Optional, PUT: Error, POST: Mandatory, DELETE: Error

14.1.6.4 Response Resource

Specific Response instance.

Sample URI: /rsps/{id1}/rsp/{id2}

Request Representation: Response
Response Representation: Response

Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

14.1.6.5 PriceResponse Resource

A specific PriceResponse instance.

Sample URI: /rsps/{id1}/rsp/{id2}

Request Representation: PriceResponse
Response Representation: PriceResponse

Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional
14.1.6.6 **TextResponse Resource**

A specific TextMessage Response instance.

Sample URI: /rps/{id1}/rsp/{id2}
Request Representation: TextResponse
Response Representation: TextResponse
Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

14.1.6.7 **DrResponse Resource**

A specific Demand Response / Load Control EndDeviceControl Response (DrResponse) instance.

Sample URI: /rps/{id1}/rsp/{id2}
Request Representation: DrResponse
Response Representation: DrResponse
Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

14.2 **Common Resources Section**

14.2.1 **Time Function Set**

14.2.1.1 **Time Resource**

Provides the Time Resource.

Sample URI: /tm
Request Representation: Time
Response Representation: Time
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

14.2.2 **Device Information Function Set**

14.2.2.1 **DeviceInformation Resource**

Device Information of the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/di
Request Representation: DeviceInformation
Response Representation: DeviceInformation
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

14.2.2.2 **SupportedLocaleList Resource**

A List of supported locales for the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/di/loc
Request Representation: SupportedLocale
Response Representation: SupportedLocaleList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

14.2.2.3 **SupportedLocale Resource**

A specific supported locale for the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/di/loc/{id2}
Request Representation: SupportedLocale
Response Representation: SupportedLocale
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory

14.2.3 Power Status Function Set

14.2.3.1 PowerStatus Resource

Contains the power status for the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/ps

Request Representation: PowerStatus

Response Representation: PowerStatus

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

14.2.4 Network Status Function Set

14.2.4.1 IPAddressList Resource

List of IPAddress instances on the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/ns

Request Representation: IPAddress

Response Representation: IPAddressList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.2 IPAddress Resource

Specific IPAddress resource. This resource may be thought of as network status information for a specific network (IP) layer interface.

Sample URI: /edev/{id1}/ns/{id2}

Request Representation: IPAddress

Response Representation: IPAddress

Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

14.2.4.3 IPAddress List Resource

List of IP Addresses

Sample URI: /edev/{id1}/ns/{id2}/addr

Request Representation: IPAddress

Response Representation: IPAddressList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.4 IPAddress Resource

A specific IP Address

Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}

Request Representation: IPAddress

Response Representation: IPAddress

Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

14.2.4.5 RPLInstanceList Resource

List of RPL instances that the IPAddress is a member

Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl

Request Representation: RPLInstance

Response Representation: RPLInstanceList
14.2.4.6 **RPLInstance Resource**

- A specific RPL Instance
- Sample URI: `/edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}`
- Request Representation: `RPLInstance`
- Response Representation: `RPLInstance`
- Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.7 **RPLSourceRoutesList Resource**

- List of RPL source routes
- Sample URI: `/edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt`
- Request Representation: `RPLSourceRoutes`
- Response Representation: `RPLSourceRoutesList`
- Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.8 **RPLSourceRoutes Resource**

- A specific RPL source route
- Sample URI: `/edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt/{id5}`
- Request Representation: `RPLSourceRoutes`
- Response Representation: `RPLSourceRoutes`
- Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.9 **LLInterfaceList Resource**

- List of Link Layer Interfaces
- Sample URI: `/edev/{id1}/ns/{id2}/ll`
- Request Representation: `LLInterface`
- Response Representation: `LLInterfaceList`
- Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.10 **LLInterface Resource**

- A specific Link Layer Interface
- Sample URI: `/edev/{id1}/ns/{id2}/ll/{id3}`
- Request Representation: `LLInterface`
- Response Representation: `LLInterface`
- Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.11 **NeighborList Resource**

- List of 802.15.4 neighbors
- Sample URI: `/edev/{id1}/ns/{id2}/ll/{id3}/nbh`
- Request Representation: `Neighbor`
- Response Representation: `NeighborList`
- Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.2.4.12 **Neighbor Resource**

- A specific 802.15.4 neighbor
14.2.5 Log/Event Log Function Set

14.2.5.1 LogEventList Resource

A List of LogEvent instances.

Sample URI: /edev/{id1}/lel

Request Representation: LogEvent

Response Representation: LogEventList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

14.2.5.2 LogEvent Resource

A specific LogEvent entry from the LogEventList.

Sample URI: /edev/{id1}/lel/{id2}

Request Representation: LogEvent

Response Representation: LogEvent

Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Mandatory

14.3 Smart Energy Resources Section

14.3.1 Configuration Resource Function Set

14.3.1.1 Configuration Resource

Contains the configuration settings of the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/cfg

Request Representation: Configuration

Response Representation: Configuration

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

14.3.1.2 PriceResponseCfgList Resource

Contains a List of price response configuration settings.

Sample URI: /edev/{id1}/prcfg

Request Representation: PriceResponseCfg

Response Representation: PriceResponseCfgList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

14.3.1.3 PriceResponseCfg Resource

Contains the price response configuration settings for this EndDevice associated with a RateComponent.

Sample URI: /edev/{id1}/prcfg/{id2}

Request Representation: PriceResponseCfg

Response Representation: PriceResponseCfg

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory
14.3.2 Software Download Function Set

14.3.2.1 FileList Resource
A list of files
Sample URI: /file
Request Representation: File
Response Representation: FileList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.2.2 File Resource
A specific file
Sample URI: /file/{id1}
Request Representation: File
Response Representation: File
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

14.3.2.3 FileStatus Resource
The file status of a particular file download for the associated EndDevice or SelfDevice.
Sample URI: /edev/{id1}/fs
Request Representation: FileStatus
Response Representation: FileStatus
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

14.3.3 Demand Response and Load Control Function Set

14.3.3.1 DemandResponseProgramList Resource
List of DemandResponseProgram instances. Devices implementing the DemandResponseProgramList resource MAY support multiple instances of DemandResponsePrograms.
Sample URI: /dr
Request Representation: DemandResponseProgram
Response Representation: DemandResponseProgramList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.3.2 DemandResponseProgram Resource
Specific DemandResponseProgram resource. This resource can be thought of as a particular DemandResponseProgram endpoint.
Sample URI: /dr/{id1}
Request Representation: DemandResponseProgram
Response Representation: DemandResponseProgram
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.3.3 ActiveEndDeviceControlList Resource
List of EndDeviceControls that are currently active.
Sample URI: /dr/{id1}/actedc
Request Representation: EndDeviceControl
Response Representation: EndDeviceControlList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.3.4 **EndDeviceControlList Resource**

List of EndDeviceControls. Devices implementing the EndDeviceControlList resource MAY support multiple EndDeviceControls.

Sample URI: /dr/{id1}/edc
Request Representation: EndDeviceControl
Response Representation: EndDeviceControlList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.3.5 **EndDeviceControl Resource**

Specific EndDeviceControl resource. This resource can be thought of as a particular Demand Response / Load Control event for a period of time.

Sample URI: /dr/{id1}/edc/{id2}
Request Representation: EndDeviceControl
Response Representation: EndDeviceControl
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.3.6 **LoadShedAvailability Resource**

Allows clients to expose their load shed availability.

Sample URI: /edev/{id1}/lsa
Request Representation: LoadShedAvailability
Response Representation: LoadShedAvailability
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

14.3.4 **Metering Function Set**

14.3.4.1 **UsagePointList Resource**

Usage point resource list. Devices implementing the UsagePointList resource MAY support multiple instances of usage points.

Sample URI: /upt
Request Representation: UsagePoint
Response Representation: UsagePointList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.4.2 **UsagePoint Resource**

Usage point instance including links to associated information.

Sample URI: /upt/{id1}
Request Representation: UsagePoint
Response Representation: UsagePoint
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Optional

14.3.4.3 **MeterReadingList Resource**

Meter Reading list including explicit URIs for each valid meter reading resource.

Sample URI: /upt/{id1}/mr
Request Representation: MeterReading
Response Representation: MeterReadingList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.4.4 **MeterReading Resource**
Meter Reading instance which contains ReadingSet and Reading resources
Sample URI: /upt/{id1}/mr/{id2}
Request Representation: MeterReading
Response Representation: MeterReading
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

14.3.4.5 **ReadingType Resource**
Meter Reading type
Sample URI: /upt/{id1}/mr/{id2}/rt
Request Representation: ReadingType
Response Representation: ReadingType
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Error

14.3.4.6 **ReadingSetList Resource**
Reading Set list
Sample URI: /upt/{id1}/mr/{id2}/rs
Request Representation: ReadingSet
Response Representation: ReadingSetList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.4.7 **ReadingSet Resource**
Reading Set instance which contains a list of Reading(s)
Sample URI: /upt/{id1}/mr/{id2}/rs/{id3}
Request Representation: ReadingSet
Response Representation: ReadingSet
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

14.3.4.8 **ReadingList Resource**
Reading list of a particular meter reading set.
Sample URI: /upt/{id1}/mr/{id2}/rs/{id3}/r
Request Representation: Reading
Response Representation: ReadingList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.4.9 **Reading Resource**
Reading instance of a particular meter reading type.
Sample URI: /upt/{id1}/mr/{id2}/rs/{id3}/r/{id4}
Request Representation: Reading
Response Representation: Reading
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional
14.3.4.10 **MirrorUsagePointList Resource**
Mirror Usage point (meter) resource list. Devices implementing the MirrorUsagePointList resource may support multiple instances of meter mirror asset.

Sample URI: /mup
Request Representation: MirrorUsagePoint
Response Representation: MirrorUsagePointList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

14.3.4.11 **MirrorUsagePoint Resource**
Mirror Usage point instance including resources it supports.

Sample URI: /mup/{id1}
Request Representation: MirrorUsagePoint (PUT), MirrorMeterReading (POST), MirrorMeterReadingList (POST)
Response Representation: MirrorUsagePoint
Methods: GET/HEAD: Optional, PUT: Mandatory, POST: Mandatory, DELETE: Mandatory

14.3.5 **Pricing Function Set**

14.3.5.1 **TariffProfileList Resource**
List of TariffProfile instances.

Sample URI: /tp
Request Representation: TariffProfile
Response Representation: TariffProfileList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.5.2 **TariffProfile Resource**
Specific TariffProfile instance. Allows clients to obtain information about the rate code.

Sample URI: /tp/{id1}
Request Representation: TariffProfile
Response Representation: TariffProfile
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.5.3 **RateComponentList Resource**
List of RateComponent instances. This list specifies a rate-specific container for the charges associated with a specific ReadingType, also referenced by the Metering function set.

Sample URI: /tp/{id1}/rc
Request Representation: RateComponent
Response Representation: RateComponentList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.5.4 **RateComponent Resource**
Specific RateComponent instance. Includes link(s) to the list of TimeTariffIntervals that apply to referenced ReadingType for the RateComponent instance.

Sample URI: /tp/{id1}/rc/{id2}
Request Representation: RateComponent
Response Representation: RateComponent
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.5.5 **ActiveTimeTariffIntervalList Resource**
The active TimeTariffInterval instance(s) for a particular TariffProfile.
Sample URI: /tp/{id1}/rc/{id2}/acttti
Request Representation: TimeTariffInterval
Response Representation: TimeTariffIntervalList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.5.6 **TimeTariffIntervalList Resource**
Collection of TimeTariffInterval instances, including associated ConsumptionTariffInterval.
Sample URI: /tp/{id1}/rc/{id2}/tti
Request Representation: TimeTariffInterval
Response Representation: TimeTariffIntervalList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.5.7 **TimeTariffInterval Resource**
Specific TimeTariffInterval instance that represents a unique usage interval and associated charges for the premises.
Sample URI: /tp/{id1}/rc/{id2}/tti/{id3}
Request Representation: TimeTariffInterval
Response Representation: TimeTariffInterval
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.5.8 **ConsumptionTariffIntervalList Resource**
List of ConsumptionTariffInterval instances.
Sample URI: /tp/{id1}/rc/{id2}/tti/{id3}/cti
Request Representation: ConsumptionTariffInterval
Response Representation: ConsumptionTariffIntervalList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.5.9 **ConsumptionTariffInterval Resource**
Specific ConsumptionTariffInterval instance.
Sample URI: /tp/{id1}/rc/{id2}/tti/{id3}/cti/{id4}
Request Representation: ConsumptionTariffInterval
Response Representation: ConsumptionTariffInterval
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.6 **Messaging Function Set**

14.3.6.1 **MessagingProgramList Resource**
List of MessagingProgram instances or channels. Devices implementing the /msg resource MAY support multiple instances of MessagingProgram
Sample URI: /msg
Request Representation: MessagingProgram
Response Representation: MessagingProgramList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.6.2 **MessagingProgram Resource**
Specific MessagingProgram instance. This resource can be thought of as a particular TextMessageList or channel.

Sample URI: /msg/{id1}
Request Representation: MessagingProgram
Response Representation: MessagingProgram
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.6.3 **ActiveTextMessageList Resource**
List of Messages that are currently active.

Sample URI: /msg/{id1}/acttxt
Request Representation: TextMessage
Response Representation: TextMessageList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.6.4 **TextMessageList Resource**
A list of TextMessages.

Sample URI: /msg/{id1}/txt
Request Representation: TextMessage
Response Representation: TextMessageList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.6.5 **TextMessage Resource**
An individual TextMessage.

Sample URI: /msg/{id1}/txt/{id2}
Request Representation: TextMessage
Response Representation: TextMessage
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.7 **Billing Function Set**

14.3.7.1 **CustomerAccountList Resource**
A List of customer accounts

Sample URI: /bill
Request Representation: CustomerAccount
Response Representation: CustomerAccountList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.2 **CustomerAccount Resource**
Customer account information

Sample URI: /bill/{id1}
Request Representation: CustomerAccount
Response Representation: CustomerAccount
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.7.3 CustomerAgreementList Resource
A list of customer agreements

Sample URI: /bill/{id1}/ca
Request Representation: CustomerAgreement
Response Representation: CustomerAgreementList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.4 CustomerAgreement Resource
A customer agreement

Sample URI: /bill/{id1}/ca/{id2}
Request Representation: CustomerAgreement
Response Representation: CustomerAgreement
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.7.5 ActiveBillingPeriodList Resource
A list of active billing periods.

Sample URI: /bill/{id1}/ca/{id2}/actbp
Request Representation: BillingPeriod
Response Representation: BillingPeriodList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.7.6 BillingPeriodList Resource
List of BillingPeriods

Sample URI: /bill/{id1}/ca/{id2}/bp
Request Representation: BillingPeriod
Response Representation: BillingPeriodList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.7 BillingPeriod Resource
Specific Billing Period information

Sample URI: /bill/{id1}/ca/{id2}/bp/{id3}
Request Representation: BillingPeriod
Response Representation: BillingPeriod
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

14.3.7.8 ProjectionReadingList Resource
A list of reading projections.

Sample URI: /bill/{id1}/ca/{id2}/pro
Request Representation: ProjectionReading
Response Representation: ProjectionReadingList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.9 ProjectionReading Resource
A specific projection reading channel
Sample URI: /bill/{id1}/ca/{id2}/pro/{id3}
Request Representation: ProjectionReading
Response Representation: ProjectionReading
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

14.3.7.10 **BillingReadingSetList Resource**
A list of billing reading sets

Sample URI: /brs
Request Representation: BillingReadingSet
Response Representation: BillingReadingSetList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.11 **BillingReadingSet Resource**
A specific billing reading set

Sample URI: /brs/{id1}
Request Representation: BillingReadingSet
Response Representation: BillingReadingSet
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.7.12 **BillingReadingList Resource**
A list of billing readings

Sample URI: /brs/{id1}/br
Request Representation: BillingReading
Response Representation: BillingReadingList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.13 **BillingReading Resource**
A specific billing reading

Sample URI: /brs/{id1}/br/{id2}
Request Representation: BillingReading
Response Representation: BillingReading
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

14.3.7.14 **TargetReadingList Resource**
A list of billing targets.

Sample URI: /bill/{id1}/ca/{id2}/tar
Request Representation: TargetReading
Response Representation: TargetReadingList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.7.15 **TargetReading Resource**
A specific target reading channel

Sample URI: /bill/{id1}/ca/{id2}/tar/{id3}
Request Representation: TargetReading
Response Representation: TargetReading
14.3.7.16 **HistoricalReadingList Resource**

A list of verified historical readings.

Sample URI: /bill/{id1}/ca/{id2}/ver

Request Representation: HistoricalReading

Response Representation: HistoricalReadingList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.7.17 **HistoricalReading Resource**

A specific historical reading channel

Sample URI: /bill/{id1}/ca/{id2}/ver/{id3}

Request Representation: HistoricalReading

Response Representation: HistoricalReading

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.8 **Prepayment Function Set**

14.3.8.1 **PrepaymentList Resource**

A List of Prepayment instances.

Sample URI: /ppy

Request Representation: Prepayment

Response Representation: PrepaymentList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.8.2 **Prepayment Resource**

A particular Prepayment instance. Provides links to the Account Balance, Credit Register, and Operation Status resources for a particular service.

Sample URI: /ppy/{id1}

Request Representation: Prepayment

Response Representation: Prepayment

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.8.3 **AccountBalance Resource**

Account Balance instance.

Sample URI: /ppy/{id1}/ab

Request Representation: AccountBalance

Response Representation: AccountBalance

Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error
14.3.8.4 PrepayOperationStatus Resource
The Operation Status for the given service. Identifies whether service should be continued. MAY also include other status information, such as low credit warning.

Sample URI: /ppy/{id1}/os
Request Representation: PrepayOperationStatus
Response Representation: PrepayOperationStatus
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

14.3.8.5 ActiveSupply InterruptionOverrideList Resource
A list of active supply interruption overrides

Sample URI: /ppy/{id1}/actsi
Request Representation: SupplyInterruptionOverride
Response Representation: SupplyInterruptionOverrideList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.8.6 SupplyInterruptionOverrideList Resource
A List of Supply Interruption Override instances.

Sample URI: /ppy/{id1}/si
Request Representation: SupplyInterruptionOverride
Response Representation: SupplyInterruptionOverrideList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.8.7 SupplyInterruptionOverride Resource
A particular Supply Interruption Override instance. This defines a period of time during which supply would not be interrupted even if available credit has been exhausted.

Sample URI: /ppy/{id1}/si/{id2}
Request Representation: SupplyInterruptionOverride
Response Representation: SupplyInterruptionOverride
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

14.3.8.8 CreditRegisterList Resource
A List of Credit Register instances. Interface for new credit transactions.

Sample URI: /ppy/{id1}/cr
Request Representation: CreditRegister
Response Representation: CreditRegisterList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.8.9 CreditRegister Resource
A particular Credit Register instance. Records a payment transaction or other credit addition.

Sample URI: /ppy/{id1}/cr/{id2}
Request Representation: CreditRegister
Response Representation: CreditRegister
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional
14.3.9  Flow Reservation Function Set

14.3.9.1  FlowReservationRequestList Resource
List of FlowReservationRequests. Devices implementing the FlowReservationRequestList resource MAY support multiple FlowReservationRequests.

Sample URI: /edev/{id1}/frq
Request Representation: FlowReservationRequest
Response Representation: FlowReservationRequestList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

14.3.9.2  FlowReservationRequest Resource
Specific FlowReservationRequest resource. This resource can be thought of as a particular reservation event request for fast charging or discharging over a period of time.

Sample URI: /edev/{id1}/frq/{id2}
Request Representation: FlowReservationRequest
Response Representation: FlowReservationRequest
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Optional

14.3.9.3  FlowReservationResponseList Resource
List of FlowReservationResponses. Devices implementing the FlowReservationResponseList resource MAY support multiple FlowReservationResponses.

Sample URI: /edev/{id1}/frp
Request Representation: FlowReservationResponse
Response Representation: FlowReservationResponseList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

14.3.9.4  FlowReservationResponse Resource
Specific FlowReservationResponse resource. This resource can be thought of as a particular reservation event response for fast charging or discharging over a period of time.

Sample URI: /edev/{id1}/frp/{id2}
Request Representation: FlowReservationResponse
Response Representation: FlowReservationResponse
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

14.3.10  Distributed Energy Resources Function Set

14.3.10.1  DERList Resource
A list of Distributed Energy Resources

Sample URI: /edev/{id1}/der
Request Representation: DER
Response Representation: DERList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.10.2  DER Resource
The information about a specific Distributed Energy Resource.

Sample URI: /edev/{id1}/der/{id2}
14.3.10.3 **AssociatedUsagePoint Resource**

The usage point associated with this DER instance.

Sample URI: /edev/{id1}/der/{id2}/upt

Request Representation: UsagePoint

Response Representation: UsagePoint

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

14.3.10.4 **AssociatedDERProgramList Resource**

The List of DERProgram instances associated with this DER.

Sample URI: /edev/{id1}/der/{id2}/derp

Request Representation: DERProgram

Response Representation: DERProgramList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.10.5 **CurrentDERProgram Resource**

The specific DER Control Program being followed by the DER.

Sample URI: /edev/{id1}/der/{id2}/cdp

Request Representation: DERProgram

Response Representation: DERProgram

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

14.3.10.6 **DERSettings Resource**

The DER settings of the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/der/{id2}/derg

Request Representation: DERSettings

Response Representation: DERSettings

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

14.3.10.7 **DERStatus Resource**

The DER status of the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/der/{id2}/ders

Request Representation: DERStatus

Response Representation: DERStatus

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

14.3.10.8 **DERAvailability Resource**

The DER availability of the associated EndDevice or SelfDevice.

Sample URI: /edev/{id1}/der/{id2}/dera

Request Representation: DERAvailability

Response Representation: DERAvailability

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error
14.3.10.9 **DERCapability Resource**

Capabilities of the DER

Sample URI: /edev/{id1}/der/{id2}/dercap

Request Representation: DERCapability

Response Representation: DERCapability

Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

14.3.10.10 **DERProgramList Resource**

List of DERProgram instances. Devices implementing the DERProgramList resource MAY support multiple instances of DERPrograms.

Sample URI: /derp

Request Representation: DERProgram

Response Representation: DERProgramList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.10.11 **DERProgram Resource**

Specific DER Control Program collection resource. This resource can be thought of as a particular DERProgram endpoint. This representation contains simple management attributes, as well as each associated resource.

Sample URI: /derp/{id1}

Request Representation: DERProgram

Response Representation: DERProgram

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

14.3.10.12 **ActiveDERControlList Resource**

List of DERControls that are currently active.

Sample URI: /derp/{id1}/actderc

Request Representation: DERControl

Response Representation: DERControlList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

14.3.10.13 **DERControlList Resource**

List of DERControls. Devices implementing the DERControlList resource MAY support multiple DERControls.

Sample URI: /derp/{id1}/derc

Request Representation: DERControl

Response Representation: DERControlList

Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.10.14 **DERControl Resource**

Specific DERControl resource. This resource can be thought of as a particular DER control event for a period of time.

Sample URI: /derp/{id1}/derc/{id2}

Request Representation: DERControl

Response Representation: DERControl
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

14.3.10.15 DefaultDERControl Resource

The DefaultDERControl resource. This resource can be thought of as the default DERControl to be used if no active DERControl event is found.

Sample URI: /derp/{id1}/dderc
Request Representation: DefaultDERControl
Response Representation: DefaultDERControl
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Error

14.3.10.16 DERCurveList Resource

A List of DER curves
Sample URI: /derp/{id1}/dc
Request Representation: DERCurve
Response Representation: DERCurveList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

14.3.10.17 DERCurve Resource

A DER curve instance
Sample URI: /derp/{id1}/dc/{id2}
Request Representation: DERCurve
Response Representation: DERCurve
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional
Appendix B – SEP 2 Model (INFORMATIVE)

Note that the XML version of the model, contained in the XML Schema definition "sep.xsd" contained in [ZB 13-0201] is normative. This section presents a human-friendly view of the information to facilitate comments on the content.

15.1 SEP 2 Package

The Smart Energy Profile 2.0 model is organized into function sets, represented by sub-packages. However, all structures are defined inside a single namespace.

Figure 15-1: Version Information

```xml
<class Version Information
SEP 2.0
tags
defaultNamespace = http://zigbee.org/sep
elementFormDefault = qualified
schemaLocation = sep.xsd
targetNamespace = http://zigbee.org/sep
trace_id =
version = 2.0.0
```

15.1.1 DeviceCapability Package

Contains definition of the objects used to convey the resources that are implemented by the publishing host.

Figure 15-2: DeviceCapability
DeviceCapability Object (FunctionSetAssignmentsBase)
Returned by the URI provided by DNS-SD, to allow clients to find the URIs to the resources in which they are interested.

15.1.2 Common Package
This package contains objects that are used in multiple function sets.

15.1.2.1 Identification Package
Contains super-classes that define the attributes common to categories of objects.

Figure 15-3: Identification

IdentifiedObject Object (Resource)
This is a root class to provide common naming attributes for all classes needing naming attributes

description attribute (String32) [0..1]
The description is a human readable text describing or naming the object.

mRID attribute (mRIDType)
The global identifier of the object.

version attribute (VersionType) [0..1]
Contains the version number of the object. See the type definition for details.

Link Object ()
Links provide a reference, via URI, to another resource.

href attribute (anyURI) «XSDattribute»
A URI reference.

List Object (Resource)
Container to hold a collection of object instances or references. See [ZB 11-0167] Design Patterns section for additional details.
all attribute (UInt16) «XSDattribute»
The number specifying "all" of the items in the list. Required on a response to a GET, ignored otherwise.

results attribute (UInt8) «XSDattribute»
Indicates the number of items in this page of results.

ListLink Object (Link)
ListLinks provide a reference, via URI, to a List.

all attribute (UInt16) «XSDattribute»
Indicates the total number of items in the referenced list.

Resource Object ()
A resource is an addressable unit of information, either a collection (List) or instance of an object (identifiedObject, or simply, Resource)

href attribute (anyURI) [0..1] «XSDattribute»
A reference to the resource address (URI). Required in a response to a GET, ignored otherwise.

RespondableIdentifiedObject Object (RespondableResource)
An IdentifiedObject to which a Response can be requested.

description attribute (String32) [0..1]
The description is a human readable text describing or naming the object.

mRID attribute (mRIDType)
The global identifier of the object.

version attribute (VersionType) [0..1]
Contains the version number of the object. See the type definition for details.

RespondableResource Object (Resource)
A Resource to which a Response can be requested.

replyTo attribute (anyURI) [0..1] «XSDattribute»
A reference to the response resource address (URI). Required on a response to a GET if responseRequired is "true".

responseRequired attribute (HexBinary8) [0..1] «XSDattribute»
Indicates whether or not a response is required upon receipt, creation or update of this resource.
Responses shall be posted to the collection specified in "replyTo".

If the resource has a deviceCategory field, devices that match one or more of the device types indicated in deviceCategory SHALL respond according to the rules listed below. If the category does not match, the device SHALL NOT respond. If the resource does not have a deviceCategory field, a device receiving the resource SHALL respond according to the rules listed below.

Value encoded as hex according to the following bit assignments, any combination is possible.
See Table 10-10 for the list of appropriate Response status codes to be sent for these purposes.

0 - End device shall indicate that message was received
1 - End device shall indicate specific response.
2 - End user / customer response is required.
All other values reserved.

**RespondableSubscribableIdentifiedObject Object** (RespondableResource)
An IdentifiedObject to which a Response can be requested.

*description attribute* (String32) [0..1]
The description is a human readable text describing or naming the object.

*mRID attribute* (mRIDType)
The global identifier of the object.

**subscribable attribute** (SubscribableType) [0..1] «XSDattribute»
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).

*version attribute* (VersionType) [0..1]
Contains the version number of the object. See the type definition for details.

**SubscribableIdentifiedObject Object** (SubscribableResource)
An IdentifiedObject to which a Subscription can be requested.

*description attribute* (String32) [0..1]
The description is a human readable text describing or naming the object.

*mRID attribute* (mRIDType)
The global identifier of the object.

*version attribute* (VersionType) [0..1]
Contains the version number of the object. See the type definition for details.

**SubscribableList Object** (SubscribableResource)
A List to which a Subscription can be requested.

*all attribute* (UInt16) «XSDattribute»
The number specifying "all" of the items in the list. Required on GET, ignored otherwise.

*results attribute* (UInt8) «XSDattribute»
Indicates the number of items in this page of results.

**SubscribableResource Object** (Resource)
A Resource to which a Subscription can be requested.

**subscribable attribute** (SubscribableType) [0..1] «XSDattribute»
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).
15.1.2.2 **Objects Package**

Contains definitions of objects used by multiple function sets.

---

**Figure 15-4: Events**

**Figure 15-5: Programs**
Error Object ()
Contains information about the nature of an error if a request could not be completed successfully.

maxRetryDuration attribute (UInt16) [0..1]
Contains the number of seconds the client SHOULD wait before retrying the request.

reasonCode attribute (UInt16)
Code indicating the reason for failure.
0 - Invalid request format
1 - Invalid request values (e.g. invalid threshold values)
2 - Resource limit reached
3 - Conditional subscription field not supported
4 - Maximum request frequency exceeded
All other values reserved

Event Object (RespondableSubscribableIdentifiedObject)
An Event indicates information that applies to a particular period of time. Events SHALL be executed relative to the time of the server, as described in the Time function set section 11.1.

creationTime attribute (TimeType)
The time at which the Event was created.

interval attribute (DateTimeInterval)
The period during which the Event applies.

EventStatus Object ()
Current status information relevant to a specific object. The Status object is used to indicate the current status of an Event. Devices can read the containing resource (e.g. TextMessage) to get the most up to date status of the event. Devices can also subscribe to a specific resource instance to get updates when any of its attributes change, including the Status object.

currentStatus attribute (UInt8)
Field representing the current status type.

0 = Scheduled

This status indicates that the event has been scheduled and the event has not yet started. The server SHALL set the event to this status when the event is first scheduled and persist until the event has become active or has been cancelled. For events with a start time less than or equal to the current time, this status SHALL never be indicated, the event SHALL start with a status of “Active”.

1 = Active

This status indicates that the event is currently active. The server SHALL set the event to this status when the event reaches its earliest Effective Start Time.

2 = Cancelled

When events are cancelled, the Status.dateTime attribute SHALL be set to the time the cancellation occurred, which cannot be in the future. The server is responsible for maintaining the cancelled event in its collection for the duration of the original event, or until the server has run out of space and needs to store a new event. Client devices SHALL be aware of Cancelled events, determine if the Cancelled event applies to them, and cancel the event immediately if applicable.

3 = Cancelled with Randomization

The server is responsible for maintaining the cancelled event in its collection for the duration of the Effective Scheduled Period. Client devices SHALL be aware of Cancelled with Randomization events, determine if the Cancelled event applies to them, and cancel the event immediately, using the larger of (absolute value of randomizeStart) and (absolute value of randomizeDuration) as the end randomization, in seconds. This Status.type SHALL NOT be used with "regular" Events, only with specializations of RandomizableEvent.

4 = Superseded

Events marked as Superseded by servers are events that may have been replaced by new events that target the same group of device types and overlap for a given period of time. Servers SHALL mark an event as Superseded at the earliest Effective Start Time of the overlapping event. Servers are responsible for maintaining the Superseded event in their collection for the duration of the Effective Scheduled Period. Client devices encountering a Superseded event SHALL terminate execution of the event immediately and commence execution of the new event immediately, unless the current time is within the start randomization window of the superseded event, in which case the client SHALL obey the start randomization of the new event. This Status.type SHALL NOT be used with TextMessage, since multiple text messages can be active.

All other values reserved.

dateTime attribute (TimeType)

The dateTime attribute will provide a timestamp of when the current status was defined. dateTime MUST be set to the time at which the status change occurred, not a time in the future or past.

potentiallySuperseded attribute (boolean)

Set to true by a server of this event if there are events that may overlap this event in time and also overlap in DeviceCategory on the same function set instance. SHALL NOT be set to true if the event is a TextMessage.
**potentiallySupersededTime attribute** (*TimeType*) [0..1]

Indicates the time that the potentiallySuperseded flag was set.

**reason attribute** (*String192*) [0..1]

The Reason attribute allows a Service provider to provide a textual explanation of the status.

**RandomizableEvent Object** (*Event*)

An Event that can indicate time ranges over which the start time and duration SHALL be randomized.

**randomizeDuration attribute** (*OneHourRangeType*) [0..1]

Number of seconds boundary inside which a random value must be selected to be applied to the associated interval duration, to avoid sudden synchronized demand changes. If related to price level changes, sign may be ignored. Valid range is -3600 to 3600. If not specified, 0 is the default.

**randomizeStart attribute** (*OneHourRangeType*) [0..1]

Number of seconds boundary inside which a random value must be selected to be applied to the associated interval start time, to avoid sudden synchronized demand changes. If related to price level changes, sign may be ignored. Valid range is -3600 to 3600. If not specified, 0 is the default.

**15.1.2.3 Types Package**

Contains definitions of reusable data types.

---

![Figure 15-7: Types](image-url)
**AccumulationBehaviourType Object** (UInt8)

0 = Not Applicable (default, if not specified)

3 = Cumulative

The sum of the previous billing period values. Note: “Cumulative” is commonly used in conjunction with “demand.” Each demand reset causes the maximum demand value for the present billing period (since the last demand reset) to accumulate as an accumulative total of all maximum demands. So instead of “zeroing” the demand register, a demand reset has the affect of adding the present maximum demand to this accumulating total.

4 = DeltaData

The difference between the value at the end of the prescribed interval and the beginning of the interval. This is used for incremental interval data.

Note: One common application would be for load profile data, another use might be to report the number of events within an interval (such as the number of equipment energizations within the specified period of time.)

6 = Indicating

As if a needle is swung out on the meter face to a value to indicate the current value. (Note: An “indicating” value is typically measured over hundreds of milliseconds or greater, or may imply a “pusher” mechanism to capture a value. Compare this to “instantaneous” which is measured over a shorter period of time.)

9 = Summation

A form of accumulation which is selective with respect to time.

Note: “Summation” could be considered a specialization of “Bulk Quantity” according to the rules of inheritance where “Summation” selectively accumulates pulses over a timing pattern, and “BulkQuantity” accumulates pulses all of the time.

12 = Instantaneous

Typically measured over the fastest period of time allowed by the definition of the metric (usually milliseconds or tens of milliseconds.) (Note: “Instantaneous” was moved to attribute #3 in 61968-9Ed2 from attribute #1 in 61968-9Ed1.)

All other values reserved.

**ApplianceLoadReductionType Object** (UInt8)

0 - Delay Appliance Load

Parameter requesting the appliance to respond by providing a moderate load reduction for the duration of a delay period. Typically referring to a “non-emergency” event in which appliances can continue operating if already in a load consuming period.

1 - Temporary Appliance Load Reduction

Parameter requesting the appliance to respond by providing an aggressive load reduction for a short time period. Typically referring to an “emergency/spinning reserve” event in which an appliance should start shedding load if currently in a load consuming period.
* Full definition of how appliances react when receiving each parameter is document in the EPA
  document - ENERGY STAR® Program Requirements, Product Specification for Residential
  Refrigerators and Freezers, Eligibility Criteria 5, Draft 2 Version 5.0.

All other values reserved.

**CommodityType Object** (UInt8)

0 = Not Applicable (default, if not specified)

1 = Electricity secondary metered value (a premises meter is typically a secondary meter)

2 = Electricity primary metered value

4 = Air

7 = NaturalGas

8 = Propane

9 = PotableWater

10 = Steam

11 = WasteWater

12 = HeatingFluid

13 = CoolingFluid

All other values reserved.

**ConsumptionBlockType Object** (UInt8)

0 = Not Applicable (default, if not specified)

1 = Block 1

2 = Block 2

3 = Block 3

4 = Block 4

5 = Block 5

6 = Block 6

7 = Block 7

8 = Block 8

9 = Block 9

10 = Block 10

11 = Block 11

12 = Block 12

13 = Block 13

14 = Block 14

15 = Block 15
All other values reserved.

**CurrencyCode Object (UInt16)**
Follows codes defined in [ISO 4217].

- 0 - Not Applicable (default, if not specified)
- 36 - Australian Dollar
- 124 - Canadian Dollar
- 840 - US Dollar
- 978 - Euro

This is not a complete list.

**DataQualifierType Object (UInt8)**

- 0 = Not Applicable (default, if not specified)
- 2 = Average
- 8 = Maximum
- 9 = Minimum
- 12 = Normal

All other values reserved.

**DateTimeInterval Object «Compound» ()**

Interval of date and time.

- **duration attribute (UInt32)**
  Duration of the interval, in seconds.

- **start attribute (TimeType)**
  Date and time of the start of the interval.

**DeviceCategoryType Object (HexBinary32)**
The Device category types defined.

Bit positions SHALL be defined as follows:

- 0 - Programmable Communicating Thermostat
- 1 - Strip Heaters
- 2 - Baseboard Heaters
- 3 - Water Heater
- 4 - Pool Pump
- 5 - Sauna
- 6 - Hot tub
- 7 - Smart Appliance
8 - Irrigation Pump
9 - Managed Commercial and Industrial (C&I) Loads
10 - Simple misc. (Residential On/Off) loads
11 - Exterior Lighting
12 - Interior Lighting
13 - Electric Vehicle
14 - Generation Systems
15 - Load Control Switch
16 - Smart Inverter
17 - EVSE
18 - RESU
19 - Energy Management System
20 - Smart Energy Module

All other values reserved.

**DstRuleType Object** (HexBinary32)

Bit map encoded rule from which is calculated the start or end time, within the current year, to
which daylight savings time offset must be applied.

The rule encoding:

- Bits 0 - 11: seconds 0 - 3599
- Bits 12 - 16: hours 0 - 23
- Bits 17 - 19: day of the week 0 = not applicable, 1 - 7 (Monday = 1)
- Bits 20 - 24: day of the month 0 = not applicable, 1 - 31
- Bits 25 - 27: operator (detailed below)
- Bits 28 - 31: month 1 - 12

Rule value of 0xFFFFFFFF means rule processing/DST correction is disabled.

The operators:

0: DST starts/ends on the Day of the Month
1: DST starts/ends on the Day of the Week that is on or after the Day of the Month
2: DST starts/ends on the first occurrence of the Day of the Week in a month
3: DST starts/ends on the second occurrence of the Day of the Week in a month
4: DST starts/ends on the third occurrence of the Day of the Week in a month
5: DST starts/ends on the forth occurrence of the Day of the Week in a month
6: DST starts/ends on the fifth occurrence of the Day of the Week in a month
7: DST starts/ends on the last occurrence of the Day of the Week in a month
An example: DST starts on third Friday in March at 1:45 AM. The rule...

Seconds: 2700
Hours: 1
Day of Week: 5
Day of Month: 0
Operator: 4
Month: 3

**FlowDirectionType Object** (UInt8)
0 = Not Applicable (default, if not specified)
1 = Forward (delivered to customer)
19 = Reverse (received from customer)
All other values reserved.

**KindType Object** (UInt8)
0 = Not Applicable (default, if not specified)
3 = Currency
8 = Demand
12 = Energy
37 = Power
All other values reserved.

**LocaleType Object** (String42)
[RFC 4646] identifier of a language-region

**mRIDType Object** (HexBinary128)
A master resource identifier. The IANA PEN [PEN] provider ID SHALL be specified in bits 0-31, the least-significant bits, and objects created by that provider SHALL be assigned unique IDs with the remaining 96 bits.

0xFFFFFFFFFFFFFFFFFFFFFF[XXXXXXXX], where [XXXXXXXX] is the PEN, is reserved for a object that is being created (e.g., a ReadingSet for the current time that is still accumulating).

Except for this special reserved identifier, each modification of an object (resource) representation MUST have a different "version".

**OneHourRangeType Object** (Int16)
A signed time offset, typically applied to a Time value, expressed in seconds, with range -3600 to 3600.

**PENType Object** (UInt32)
IANA Private Enterprise Number [PEN].
**PerCent Object** (UInt16)

Used for percentages, specified in hundredths of a percent, 0 - 10000. (10000 = 100%)

**PhaseCode Object** (UInt8)

0 = Not Applicable (default, if not specified)

32 = Phase C (and S2)

33 = Phase CN (and S2N)

40 = Phase CA

64 = Phase B

65 = Phase BN

66 = Phase BC

128 = Phase A (and S1)

129 = Phase AN (and S1N)

132 = Phase AB

224 = Phase ABC

All other values reserved.

**PINType Object** (UInt32)

6 digit unsigned decimal integer (0 - 999999).

(Note that this only requires 20 bits, if it can be allocated.)

**PowerOfTenMultiplierType Object** (Int8)

-9 = nano=x10^-9

-6 = micro=x10^-6

-3 = milli=x10^-3

0 = none=x1 (default, if not specified)

1 = deca=x10

2 = hecto=x100

3 = kilo=x1000

6 = Mega=x10^6

9 = Giga=x10^9

This is not a complete list. Any integer between -9 and 9 SHALL be supported, indicating the
power of ten multiplier for the units.

**PrimacyType Object** (UInt8)

Values possible for indication of "Primary" provider:

0: In home energy management system

1: Contracted premises service provider
2: Non-contractual service provider

All other values reserved.

**RealEnergy Object** ()

Real electrical energy

*multiplier attribute* (*PowerOfTenMultiplierType*)

Multiplier for 'unit'.

*value attribute* (*UInt48*)

Value of the energy in Watt-hours. (uom 72)

**RoleFlagsType Object** (*HexBinary16*)

Specifies the roles that apply to a usage point.

Bit 0 - *isMirror* - SHALL be set if the server is not the measurement device

Bit 1 - *isPremisesAggregationPoint* - SHALL be set if the UsagePoint is the point of delivery for a premises

Bit 2 - *isPEV* - SHALL be set if the usage applies to an electric vehicle

Bit 3 - *isDER* - SHALL be set if the usage applies to a distributed energy resource, capable of delivering power to the grid.

Bit 4 - *isRevenueQuality* - SHALL be set if usage was measured by a device certified as revenue quality

Bit 5 - *isDC* - SHALL be set if the usage point measures direct current

Bit 6 - *isSubmeter* - SHALL be set if the usage point is not a premises aggregation point

Bit 7-15 - Reserved

**ServiceKind Object** (*UInt8*)

Service kind

0 - electricity

1 - gas

2 - water

3 - time

4 - pressure

5 - heat

6 - cooling

All other values reserved.

**SFDIType Object** (*UInt40*)

Unsigned integer, max inclusive 687194767359, which is $2^{36} - 1$ (68719476735), with added check digit. See Section 8.3.2 for check digit calculation.

**SignedPerCent Object** (*Int16*)

Used for signed percentages, specified in hundredths of a percent, -10000 - 10000. (10000 =
100%

SignedRealEnergy Object ()
Real electrical energy, signed.

multiplier attribute (PowerOfTenMultiplierType)
Multiplier for 'unit'.

value attribute (Int48)
Value of the energy in Watt-hours. (uom 72)

SubscribableType Object »XSDsimpleType» (UInt8)
The subscribable values.

0 - Resource does not support subscriptions
1 - Resource supports non-conditional subscriptions
2 - Resource supports conditional subscriptions
3 - Resource supports both conditional and non-conditional subscriptions
All other values reserved.

TimeOffsetType Object (Int32)
A signed time offset, typically applied to a Time value, expressed in seconds.

TimeType Object (Int64)
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)
0 = Not Applicable (default, if not specified)
1 = TOU A
2 = TOU B
3 = TOU C
4 = TOU D
5 = TOU E
6 = TOU F
7 = TOU G
8 = TOU H
9 = TOU I
10 = TOU J
11 = TOU K
12 = TOU L
13 = TOU M
14 = TOU N
15 = TOU O

All other values reserved.

**UnitType Object** (UInt8)
The unit types defined for end device control target reductions.

0 - kWh
1 - kW
2 - Watts
3 - Cubic Meters
4 - Cubic Feet
5 - US Gallons
6 - Imperial Gallons
7 - BTUs
8 - Liters
9 - kPA (gauge)
10 - kPA (absolute)
11 - Mega Joule
12 - Unitless

All other values reserved.

**UnitValueType Object** ()
Type for specification of a specific value, with units and power of ten multiplier.

*multiplier attribute* (*PowerOfTenMultiplierType*)
Multiplier for 'unit'.

*unit attribute* (*UomType*)
Unit in symbol

*value attribute* (*Int32*)
Value in units specified

**UomType Object** (UInt8)
0 = Not Applicable (default, if not specified)
5 = A (Current in Amperes (RMS))
6 = Kelvin (Temperature)
23 = Degrees Celsius (Relative temperature)
29 = Voltage
31 = J (Energy joule)
33 = Hz (Frequency)
38 = W (Real power in Watts)
42 = m³ (Cubic Meter)
61 = VA (Apparent power)
63 = var (Reactive power)
65 = CosTheta (Displacement Power Factor)
67 = V² (Volts squared)
69 = A² (Amp squared)
71 = VAh (Apparent energy)
72 = Wh (Real energy in Watt-hours)
73 = varh (Reactive energy)
106 = Ah (Ampere-hours / Available Charge)
119 = ft³ (Cubic Feet)
122 = ft³/h (Cubic Feet per Hour)
125 = m³/h (Cubic Meter per Hour)
128 = US gl (US Gallons)
129 = US gl/h (US Gallons per Hour)
130 = IMP gl (Imperial Gallons)
131 = IMP gl/h (Imperial Gallons per Hour)
132 = BTU
133 = BTU/h
134 = Liter
137 = L/h (Liters per Hour)
140 = PA(gauge)
155 = PA(absolute)
169 = Therm
All other values reserved.

VersionType Object (UInt16)
Version SHALL indicate a distinct identifier for each revision of an IdentifiedObject. If not
specified, a default version of "0" (initial version) SHALL be assumed. Upon modification of any
IdentifiedObject, the mRID SHALL remain the same, but the version SHALL be incremented.
Servers MAY NOT modify objects that they did not create, unless they were notified of the
change from the entity controlling the object's PEN.
15.1.2.4 Primitive Types Package

Contains definitions of primitive data types based on XML schema primitives.

**Figure 15-8: Primitive Types**

- **HexBinary8 Object**: XSDsimpleType (hexBinary)
  - An 8-bit field encoded as a hex string (2 hex characters). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0".

- **HexBinary16 Object**: XSDsimpleType (hexBinary)
  - A 16-bit field encoded as a hex string (4 hex characters max). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0".

- **HexBinary32 Object**: XSDsimpleType (hexBinary)
  - A 32-bit field encoded as a hex string (8 hex characters max). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0".

- **HexBinary48 Object**: XSDsimpleType (hexBinary)
  - A 48-bit field encoded as a hex string (12 hex characters max). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0".

- **HexBinary64 Object**: XSDsimpleType (hexBinary)
  - A 64-bit field encoded as a hex string (16 hex characters max). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0".

- **HexBinary128 Object**: XSDsimpleType (hexBinary)
  - A 128-bit field encoded as a hex string (32 hex characters max). Where applicable, bit 0, or the least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an odd number of characters requires a leading "0".
HexBinary160 Object «XSDsimpleType» (hexBinary)
A 160-bit field encoded as a hex string (40 hex characters max). Where applicable, bit 0, or the
least significant bit, goes on the right. Note that hexBinary requires pairs of hex characters, so an
odd number of characters requires a leading "0".

String6 Object «XSDsimpleType» (string)
Character string of max length 6. In order to limit internal storage, implementations SHALL
reduce the length of strings using multi-byte characters so that the string may be stored using
"maxLength" octets in the given encoding.

String16 Object «XSDsimpleType» (string)
Character string of max length 16. In order to limit internal storage, implementations SHALL
reduce the length of strings using multi-byte characters so that the string may be stored using
"maxLength" octets in the given encoding.

String20 Object «XSDsimpleType» (string)
Character string of max length 20. In order to limit internal storage, implementations SHALL
reduce the length of strings using multi-byte characters so that the string may be stored using
"maxLength" octets in the given encoding.

String32 Object «XSDsimpleType» (string)
Character string of max length 32. In order to limit internal storage, implementations SHALL
reduce the length of strings using multi-byte characters so that the string may be stored using
"maxLength" octets in the given encoding.

String42 Object «XSDsimpleType» (string)
Character string of max length 42. In order to limit internal storage, implementations SHALL
reduce the length of strings using multi-byte characters so that the string may be stored using
"maxLength" octets in the given encoding.

String192 Object «XSDsimpleType» (string)
Character string of max length 192. For all string types, in order to limit internal storage,
implementations SHALL reduce the length of strings using multi-byte characters so that the
string may be stored using "maxLength" octets in the given encoding.

UInt8 Object «XSDsimpleType» (unsignedByte)
Unsigned integer, max inclusive 255 (2^8-1)

UInt16 Object «XSDsimpleType» (unsignedShort)
Unsigned integer, max inclusive 65535 (2^16-1)

UInt32 Object «XSDsimpleType» (unsignedInt)
Unsigned integer, max inclusive 4294967295 (2^32-1)

UInt40 Object «XSDsimpleType» (unsignedLong)
Unsigned integer, max inclusive 1099511627775 (2^40-1)

UInt48 Object «XSDsimpleType» (unsignedLong)
Unsigned integer, max inclusive 281474976710655 (2^48-1)

UInt64 Object «XSDsimpleType» (unsignedLong)
Unsigned integer, max inclusive 18446744073709551615 (2^64-1)
15.1.3 **EndDevice Package**

![Diagram of EndDevice Package](image)

**Figure 15-9: SelfDevice**
Figure 15-10: EndDevice

AbstractDevice Object (SubscribableResource)
The EndDevice providing the resources available within the DeviceCapabilities.

loadShedDeviceCategory attribute (DeviceCategoryType) [0..1]
This field is for use in devices that can shed load. If you are a device that does not respond to EndDeviceControls (for instance, an ESI), this field should not have any bits set.

sFDI attribute (SFDIType)
Short form of device identifier, WITH the checksum digit. See the Security section for additional details.

DeviceStatus Object (Resource)
Status of device

changedTime attribute (TimeType)
The time at which the reported values were recorded.

onCount attribute (UInt16) [0..1]
The number of times that the device has been turned on: Count of "device on" times, since the last time the counter was reset

opState attribute (UInt8) [0..1]
Device operational state:

0 - Not applicable / Unknown
1 - Not operating
2 - Operating
3 - Starting up
4 - Shuting down
5 - At disconnect level
6 - kW ramping
7 - kVar ramping

**opTime attribute** *(UInt32) [0..1]*
Total time device has operated: re-settable: Accumulated time in seconds since the last time the counter was reset.

**EndDevice Object** *(AbstractDevice)*
Asset container that performs one or more end device functions. Contains information about individual devices in the network.

**EndDeviceList Object** *(SubscribableList)*
A List element to hold EndDevice objects.

**Registration Object** *(Resource)*
Registration represents an authorization to access the resources on a host.

**dateTimeRegistered attribute** *(TimeType)*
Contains the time at which this registration was created, by which clients MAY prioritize information providers with the most recent registrations, when no additional direction from the consumer is available.

**pIN attribute** *(PINType)*
Contains the registration PIN number associated with the device, including the checksum digit.

**SelfDevice Object** *(AbstractDevice)*
The EndDevice providing the resources available within the DeviceCapabilities.

**Temperature Object** ()
Specification of a temperature.

**multiplier attribute** *(PowerOfTenMultiplierType)*
Multiplier for 'unit'.

**subject attribute** *(UInt8)*
The subject of the temperature measurement
0 - Enclosure
1 - Transformer
2 - HeatSink

**value attribute** *(Int16)*
Value in Degrees Celsius (uom 23).
15.1.4 FunctionSetAssignments Package

FunctionSetAssignmentsBase Object (Resource)
Defines a collection of function set instances that are to be used by one or more devices as indicated by the EndDevice object(s) of the server.

FunctionSetAssignments Object (FunctionSetAssignmentsBase)
Provides an identifiable, subscribable collection of resources for a particular device to consume.

\begin{itemize}
  \item \textit{description attribute (String32) [0..1]}
  \item \textit{mRID attribute (mRIDType)}
  \item \textit{subscribable attribute (SubscribableType) [0..1] «XSDattribute»}
  \item \textit{version attribute (VersionType) [0..1]}
\end{itemize}

The description is a human readable text describing or naming the object.

The global identifier of the object.

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).

Contains the version number of the object. See the type definition for details.

FunctionSetAssignmentsList Object (SubscribableList)
A List element to hold FunctionSetAssignments objects.
15.1.5 **Pub-Sub Package**

Contains resource definitions used to allow subscriptions and notifications of publications.

**Condition Object**

Indicates a condition that must be satisfied for the Notification to be triggered.

- **attributeIdentifier attribute** (*UInt8*)
  - 0 = Reading value
  - 1-255 = Reserved

- **lowerThreshold attribute** (*Int48*)
  - The value of the lower threshold

- **upperThreshold attribute** (*Int48*)
  - The value of the upper threshold

**SubscriptionBase Object** *(Resource)*

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.

- **subscribedResource attribute** (*anyURI*)
  - 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.

**Subscription Object** *(SubscriptionBase)*

Holds the information related to a client subscription to receive updates to a resource
automatically.

**encoding attribute** *(UInt8)*

0 - application/sep+xml
1 - application/sep-exi
2-255 - reserved

**level attribute** *(String16)*

Contains the preferred schema and extensibility level indication such as "+S0"

**limit attribute** *(UInt16)*

This element is used to indicate the maximum number of list items that should be included in a notification when the subscribed resource changes. This limit is meant to be functionally equivalent to the ‘limit’ query string parameter, but applies to both list resources as well as other resources. For list resources, if a limit of ‘0’ is specified, then notifications SHALL contain a list resource with results='0' (equivalent to a simple change notification). For list resources, if a limit greater than ‘0’ is specified, then notifications SHALL contain a list resource with results equal to the limit specified (or less, should the list contain fewer items than the limit specified or should the server be unable to provide the requested number of items for any reason) and follow the same rules for list resources (e.g., ordering). For non-list resources, if a limit of ‘0’ is specified, then notifications SHALL NOT contain a resource representation (equivalent to a simple change notification). For non-list resources, if a limit greater than ‘0’ is specified, then notifications SHALL contain the representation of the changed resource.

**notificationURI attribute** *(anyURI)*

The resource to which to post the notifications about the requested subscribed resource. Because this URI will exist on a server other than the one being POSTed to, this attribute SHALL be a fully-qualified absolute URI, not a relative reference.

**SubscriptionList Object** *(List)*

A List element to hold Subscription objects.

**Notification Object** *(SubscriptionBase)*

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.

**newResourceURI attribute** *(anyURI)* [0..1]

The new location of the resource, if moved.

**status attribute** *(UInt8)*

0 = Default Status
1 = Subscription canceled, no additional information
2 = Subscription canceled, resource moved
3 = Subscription canceled, resource definition changed (e.g., SEP 2.0 to 2.1)
4 = Subscription canceled, resource deleted
All other values reserved.

**subscriptionURI attribute** *(anyURI)*

The subscription from which this notification was triggered.
**NotificationList Object** (List)

A List element to hold Notification objects.

**15.1.6 Response Package**

Contains definitions of objects enabling responses to be sent back to suppliers and providers.

---

**AppliedTargetReduction Object** ()

Specifies the value of the TargetReduction applied by the device.

- **type attribute** (*UnitType*)
  - Enumerated field representing the type of reduction requested.

- **value attribute** (*UInt16*)
  - Indicates the requested amount of the relevant commodity to be reduced.

**DrResponse Object** (Response)

A response to a Demand Response Load Control (EndDeviceControl) message.

- **overrideDuration attribute** (*UInt16*) [0..1]
  - 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.

**PriceResponse Object** (Response)

A response related to a price message.

---

**Figure 15-13: Response**
Response Object (Resource)
The Response object is the generic response data repository for functions which do not have
additional specific data (e.g. DRLC has additional data fields (SetPoint) where Price and Text
event do not).

   createdDateTime attribute (TimeType) [0..1]
The createdDateTime field contains the date and time when the acknowledgement/status occurred in the
client. The client will provide the timestamp to ensure the proper time is captured in case the response is
delayed in reaching the server (server receipt time would not be the same as the actual confirmation time).
The time reported from the client should be relative to the time server indicated by the
FunctionSetAssignment that also indicated the event resource; if no FunctionSetAssignment exists, the
time of the server where the event resource was hosted.

   endDeviceLFDI attribute (HexBinary160)
Contains the LFDI of the device providing the response.

   status attribute (UInt8) [0..1]
The status field contains the acknowledgement or status. Each event type (DR/LC, Price, or Text) can
return different status information (e.g. an Acknowledge will be returned for a Price event where a DRLC
event can return Event Received, Event Started, and Event Completed). The Status field value definitions
are defined in Table 10-10 Response Types by Function Set.

   subject attribute (mRIDType)
The subject field provides a method to match the response with the originating event. It is populated with
the mRID of the original object.

ResponseList Object (List)
A List element to hold Response objects.

ResponseSet Object (IdentifiedObject)
A container for a ResponseList.

ResponseSetList Object (List)
A List element to hold ResponseSet objects.

TextResponse Object (Response)
A response to a text message
15.1.7 **Time Package**

**class Time**

<table>
<thead>
<tr>
<th>Resource</th>
<th>FunctionSetAssignmentsBase</th>
<th>DeviceStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Link</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td>TimeLink</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Resource**

- currentTime : TimeType
- dstEndTime : TimeType
- dstOffset : TimeOffsetType
- dstStartTime : TimeType
- localTime : TimeType [0..1]
- quality : UInt8
- tzOffset : TimeOffsetType

**Time**

**currentTime** *(TimeType)*

The current time, in the format defined by TimeType.

**dstEndTime** *(TimeType)*

Time at which daylight savings ends (dstOffset no longer applied). Result of dstEndRule calculation.

**dstOffset** *(TimeOffsetType)*

Daylight savings time offset from local standard time. A typical practice is advancing clocks one hour when daylight savings time is in effect, which would result in a positive dstOffset.

**dstStartTime** *(TimeType)*

Time at which daylight savings begins (apply dstOffset). Result of dstStartRule calculation.

**localTime** *(TimeType)* [0..1]

Local time: localTime = currentTime + tzOffset (+ dstOffset when in effect).

**quality** *(UInt8)*

Metric indicating the quality of the time source from which the service acquired time. Lower (smaller) quality enumeration values are assumed to be more accurate.

- 3 - time obtained from external authoritative source such as NTP
- 4 - time obtained from level 3 source
- 5 - time manually set or obtained from level 4 source
- 6 - time obtained from level 5 source
- 7 - time intentionally uncoordinated

All other values are reserved for future use.

**notes**

*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.*
**tzOffset attribute (TimeOffsetType)**

Local time zone offset from currentTime. Does not include any daylight savings time offsets. For American time zones, a negative tzOffset SHALL be used (eg. EST = GMT-5 which is -18000).

15.1.8 **DeviceInformation Package**

Contains general information about devices.

**DeviceInformation Object** (Resource)

Contains identification and other information about the device that changes very infrequently, typically only when updates are applied, if ever.

**functionsImplemented attribute (HexBinary64) [0..1]**

Bitmap indicating the function sets used by the device as a client.

0 - Device Capability
1 - Self Device Resource
2 - End Device Resource
3 - Function Set Assignments
4 - Subscription/Notification Mechanism
5 - Response
6 - Time

---

**Figure 15-15: DeviceInformation**

![Diagram of DeviceInformation](image)
7 - Device Information
8 - Power Status
9 - Network Status
10 - Log/Event Log
11 - Configuration Resource
12 - Software Download
13 - DRLC
14 - Metering
15 - Pricing
16 - Messaging
17 - Billing
18 - Prepayment
19 - Flow Reservation
20 - DER Control

**lFDI attribute** (*HexBinary160*)
Long form device identifier. See the Security section for full details.

**mfDate attribute** (*TimeType*)
Date/time of manufacture

**mfHwVer attribute** (*String32*)
Manufacturer hardware version

**mfID attribute** (*PENType*)
The manufacturer's IANA Enterprise Number.

**mfInfo attribute** (*String32*) [0..1]
Manufacturer dependent information related to the manufacture of this device

**mfModel attribute** (*String32*)
Manufacturer's model number

**mfSerNum attribute** (*String32*)
Manufacturer assigned serial number

**primaryPower attribute** (*PowerSourceType*)
Primary source of power.

**secondaryPower attribute** (*PowerSourceType*)
Secondary source of power

**swActTime attribute** (*TimeType*)
Activation date/time of currently running software

**swVer attribute** (*String32*)
Currently running software version
DRLCCapabilities Object ()
Contains information about the static capabilities of the device, to allow service providers to
know what types of functions are supported, what the normal operating ranges and limits are, and
other similar information, in order to provide better suggestions of applicable programs to receive
the maximum benefit.

averageEnergy attribute (RealEnergy)
The average hourly energy usage when in normal operating mode.

maxDemand attribute (ActivePower)
The maximum demand rating of this end device.

optionsImplemented attribute (HexBinary32)
Bitmap indicating the DRLC options implemented by the device.

0 - Target reduction (kWh)
1 - Target reduction (kW)
2 - Target reduction (Watts)
3 - Target reduction (Cubic Meters)
4 - Target reduction (Cubic Feet)
5 - Target reduction (US Gallons)
6 - Target reduction (Imperial Gallons)
7 - Target reduction (BTUs)
8 - Target reduction (Liters)
9 - Target reduction (kPA (gauge))
10 - Target reduction (kPA (absolute))
11 - Target reduction (Mega Joule)
12 - Target reduction (Unitless)
13-15 - Reserved
16 - Temperature set point
17 - Temperature offset
18 - Duty cycle
19 - Load adjustment percentage
20 - Appliance load reduction
21-32 - Reserved

SupportedLocale Object (Resource)
Specifies a locale that is supported

locale attribute (LocaleType)
The code for a locale that is supported
15.1.9 **PowerStatus Package**

**SupportedLocaleList Object** (List)
A List element to hold SupportedLocale objects.

**PowerStatus Object** (Resource)
Contains the status of the device's power sources

* batteryStatus attribute (*UInt8*)
Battery system status

0 = unknown
1 = normal (more than LowChargeThreshold remaining)
2 = low (less than LowChargeThreshold remaining)
3 = depleted (0% charge remaining)
4 = not applicable (mains powered only)

* changedTime attribute (*TimeType*)
The time at which the reported values were recorded.

* currentPowerSource attribute (*PowerSourceType*)
This value will be fixed for devices powered by a single source. This value may change for devices able to transition between multiple power sources (mains to battery backup, etc.).

* estimatedChargeRemaining attribute (*PerCent*) [0..1]
Estimate of remaining battery charge as a percent of full charge.

* estimatedTimeRemaining attribute (*UInt32*) [0..1]
Estimated time (in seconds) to total battery charge depletion (under current load)
**sessionTimeOnBattery** attribute (UInt32) [0..1]

If the device has a battery, this is the time since the device last switched to battery power, or the time since the device was restarted, whichever is less, in seconds.

**totalTimeOnBattery** attribute (UInt32) [0..1]

If the device has a battery, this is the total time the device has been on battery power, in seconds. It may be reset when the battery is replaced.

**PowerSourceType Object** (UInt8)
0 - none
1 - mains
2 - battery
3 - local generation
4 - emergency
5 - unknown
All other values reserved.

**PEVInfo Object** ()
Contains attributes that can be exposed by PEVs and other devices that have charging requirements.

**chargingPowerNow** attribute (ActivePower)
This is the actual power flow in or out of the charger or inverter. This is calculated by the vehicle based on actual measurements. This number is positive for charging.

**energyRequestNow** attribute (RealEnergy)
This is the amount of energy that must be transferred from the grid to EVSE and PEV to achieve the target state of charge allowing for charger efficiency and any vehicle and EVSE parasitic loads. This is calculated by the vehicle and changes throughout the connection as forward or reverse power flow change the battery state of charge. This number is positive for charging.

**maxForwardPower** attribute (ActivePower)
This is maximum power transfer capability that could be used for charging the PEV to perform the requested energy transfer. It is the lower of the vehicle or EVSE physical power limitations. It is not based on economic considerations. The vehicle may draw less power than this value based on its charging cycle. The vehicle defines this parameter. This number is positive for charging power flow.

**minimumChargingDuration** attribute (UInt32)
This is computed by the PEV based on the charging profile to complete the energy transfer if the maximum power is authorized. The value will never be smaller than the ratio of the energy request to the power request because the charging profile may not allow the maximum power to be used throughout the transfer. This is a critical parameter for determining whether any slack time exists in the charging cycle between the current time and the TCIN.

**targetStateOfCharge** attribute (PerCent)
This is the target state of charge that is to be achieved during charging before the time of departure (TCIN). The default value is 100%. The value cannot be set to a value less than the actual state of charge.

**timeChargelsNeeded** attribute (TimeType)
Time Charge is Needed (TCIN) is the time that the PEV is expected to depart. The value is manually entered using controls and displays in the vehicle or on the EVSE or using a mobile device. It is 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.

15.1.10 **NetworkStatus Package**

- **IEEE_802_15_4 Object**
  - Contains 802.15.4 link layer specific attributes.
    - **capabilityInfo attribute (HexBinary8)**
    - As defined by IEEE 802.15.4
    - **shortAddress attribute (UInt16)**

---

**Figure 15-17: NetworkStatus**
As defined by IEEE 802.15.4

**IPAddr Object** (Resource)
An Internet Protocol address object.

*address attribute* *(HexBinary128)*
An IP address value.

**IPAddrList Object** (List)
List of IPAddr instances.

**IPInterface Object** (Resource)
Specific IPInterface resource. This resource may be thought of as network status information for a specific network (IP) layer interface.

*ifDescr attribute* *(String192) [0..1]*
Use rules from [RFC 2863].

*ifHighSpeed attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863].

*ifInBroadcastPkts attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863].

*ifIndex attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863].

*ifInDiscards attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863]. Can be thought of as Input Datagrams Discarded.

*ifInErrors attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863].

*ifInMulticastPkts attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863]. Can be thought of as Multicast Datagrams Received.

*ifInOctets attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863]. Can be thought of as Bytes Received.

*ifInUcastPkts attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863]. Can be thought of as Datagrams Received.

*ifInUnknownProtos attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863]. Can be thought of as Datagrams with Unknown Protocol Received.

*ifMtu attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863].

*ifName attribute* *(String16) [0..1]*
Use rules from [RFC 2863].

*ifOperStatus attribute* *(UInt8) [0..1]*
Use rules and assignments from [RFC 2863].

*ifOutBroadcastPkts attribute* *(UInt32) [0..1]*
Use rules from [RFC 2863]. Can be thought of as Broadcast Datagrams Sent.
ifOutDiscards attribute (UInt32) [0..1]
Use rules from [RFC 2863]. Can be thought of as Output Datagrams Discarded.

ifOutErrors attribute (UInt32) [0..1]
Use rules from [RFC 2863].

ifOutMulticastPkts attribute (UInt32) [0..1]
Use rules from [RFC 2863]. Can be thought of as Multicast Datagrams Sent.

ifOutOctets attribute (UInt32) [0..1]
Use rules from [RFC 2863]. Can be thought of as Bytes Sent.

ifOutUcastPkts attribute (UInt32) [0..1]
Use rules from [RFC 2863]. Can be thought of as Datagrams Sent.

ifPromiscuousMode attribute (boolean) [0..1]
Use rules from [RFC 2863].

ifSpeed attribute (UInt32) [0..1]
Use rules from [RFC 2863].

ifType attribute (UInt16) [0..1]
Use rules and assignments from [RFC 2863].

lastResetTime attribute (Int64) [0..1]
Similar to ifLastChange in [RFC 2863].

lastUpdatedTime attribute (Int64) [0..1]
The date/time of the reported status.

IPInterfaceList Object (List)
List of IPInterface instances.

LLInterface Object (Resource)
A link-layer interface object.

CRCerrors attribute (UInt32)
Contains the number of CRC errors since reset.

EUI64 attribute (HexBinary64)
Contains the EUI-64 of the link layer interface. 48 bit MAC addresses SHALL be changed into an EUI-64 using the method defined in [RFC 4291], Appendix A. (The method is to insert "0xFFFE" as described in the reference.)

linkLayerType attribute (UInt8)
Specifies the type of link layer interface associated with the IPInterface. Values are below.
0 = Unspecified
1 = IEEE 802.3 (Ethernet)
2 = IEEE 802.11 (WLAN)
3 = IEEE 802.15 (PAN)
4 = IEEE 1901 (PLC)
All other values reserved.
**LLAckNotRx attribute** (UInt32) [0..1]
Number of times an ACK was not received for a frame transmitted (when ACK was requested).

**LLCSMAFail attribute** (UInt32) [0..1]
Number of times CSMA failed.

**LLFramesDropRx attribute** (UInt32) [0..1]
Number of dropped receive frames.

**LLFramesDropTx attribute** (UInt32) [0..1]
Number of dropped transmit frames.

**LLFramesRx attribute** (UInt32) [0..1]
Number of link layer frames received.

**LLFramesTx attribute** (UInt32) [0..1]
Number of link layer frames transmitted.

**LLMediaAccessFail attribute** (UInt32) [0..1]
Number of times access to media failed.

**LLOctetsRx attribute** (UInt32) [0..1]
Number of Bytes received.

**LLOctetsTx attribute** (UInt32) [0..1]
Number of Bytes transmitted.

**LLRetryCount attribute** (UInt32) [0..1]
Number of MAC transmit retries.

**LLSecurityErrorRx attribute** (UInt32) [0..1]
Number of receive security errors.

**LLInterfaceList Object** (List)
List of LLInterface instances.

**IoWPAN Object** ()
Contains information specific to 6LoWPAN.

**octetsRx attribute** (UInt32) [0..1]
Number of Bytes received

**octetsTx attribute** (UInt32) [0..1]
Number of Bytes transmitted

**packetsRx attribute** (UInt32)
Number of packets received

**packetsTx attribute** (UInt32)
Number of packets transmitted

**rxFragError attribute** (UInt32)
Number of errors receiving fragments

**Neighbor Object** (Resource)
Contains 802.15.4 link layer specific attributes.
isChild attribute (boolean)

True if the neighbor is a child.

linkQuality attribute (UInt8)

The quality of the link, as defined by 802.15.4

shortAddress attribute (UInt16)

As defined by IEEE 802.15.4

NeighborList Object (List)

List of 15.4 neighbors.

RPLInstance Object (Resource)

Specific RPLInstance resource. This resource may be thought of as network status information for a specific RPL instance associated with IPInterface.

DODAGid attribute (UInt8)

See [RFC 6550].

DODAGroot attribute (boolean)

See [RFC 6550].

flags attribute (UInt8)

See [RFC 6550].

groundedFlag attribute (boolean)

See [RFC 6550].

MOP attribute (UInt8)

See [RFC 6550].

PRF attribute (UInt8)

See [RFC 6550].

rank attribute (UInt16)

See [RFC 6550].

RPLInstanceID attribute (UInt8)

See [RFC 6550].

versionNumber attribute (UInt8)

See [RFC 6550].

RPLInstanceList Object (List)

List of RPLInstances associated with the IPInterface.

RPLSourceRoutes Object (Resource)

A RPL source routes object.

DestAddress attribute (HexBinary128)

See [RFC 6554].

SourceRoute attribute (HexBinary128)

See [RFC 6554].

RPLSourceRoutesList Object (List)

List of RPL source routes if the hosting device is the DODAGroot
15.1.11 LogEvents Package

A time stamped instance of a significant event detected by the device.

**createdDateTime attribute** (*TimeType*)

The date and time that the event occurred.

**extendedData attribute** (*UInt32*) [0..1]

May be used to transmit additional details about the event.

**functionSet attribute** (*UInt8*)

If the profileID indicates this is the Smart Energy Profile, the functionSet is defined by the Zigbee Alliance and SHALL be one of the values from the table below (Smart Energy Profile 2.0 function set identifiers). If the profileID is anything else, the functionSet is defined by the identified profile.

- 0 General (not specific to a function set)
- 1 Publish and Subscribe

Diagram:

![LogEvents Diagram](image-url)
2 End Device
3 Function Set Assignment
4 Response
5 Demand Response and Load Control
6 Metering
7 Pricing
8 Messaging
9 Billing
10 Prepayment
11 Distributed Energy Resources
12 Time
13 Software Download
14 Device Information
15 Power Status
16 Network Status
17 Log Event List
18 Configuration
19 Security

All other values are reserved.

**logEventCode attribute** *(UInt8)*

An 8 bit unsigned integer. logEventCodes are scoped to a profile and a function set. If the profile is Smart Energy, the logEventCode is defined by the Zigbee Alliance within one of the function sets of Smart Energy Profile 2.0. If the profile is anything else, the logEventCode is defined by the specified profile.

**logEventID attribute** *(UInt16)*

This 16-bit value, combined with createdDateTime, profileID, and logEventPEN, should provide a reasonable level of uniqueness.

**logEventPEN attribute** *(PENType)*

The Private Enterprise Number(PEN) of the entity that defined the profileID, functionSet, and logEventCode of the logEvent. ZigBee-assigned logEventCodes SHALL use the ZigBee Alliance PEN. Combinations of profileID, functionSet, and logEventCode SHALL have unique meaning within a logEventPEN and are defined by the owner of the PEN.

**profileID attribute** *(UInt8)*

The profileID identifies which profile (HA, BA, SE, etc) defines the following event information.

- 0 Not profile specific.
- 1 Vendor Defined
All other values are reserved.

**LogEventList Object** *(SubscribableList)*
A List element to hold LogEvent objects.

15.1.12 **Configuration Package**

---

**Configuration Object** *(SubscribableResource)*
This resource contains various settings to control the operation of the device

- **currentLocale attribute** *(LocaleType)*
  [RFC 4646] identifier of the language-region currently in use.

- **userDeviceName attribute** *(String32)*
  User assigned, convenience name used for network browsing displays, etc. Example "My Thermostat"

**PowerConfiguration Object**
Contains configuration related to the device's power sources

- **batteryInstallTime attribute** *(TimeType)* [0..1]
  Time/Date at which battery was installed,

- **lowChargeThreshold attribute** *(UInt32)* [0..1]
In context of the PowerStatus resource, this is the value of EstimatedTimeRemaining below which BatteryStatus "low" is indicated and the LE_LOW_BATTERY is raised.

**PriceResponseCfg Object** (Resource)
Configuration data that specifies how price responsive devices SHOULD respond to price changes while acting upon a given RateComponent.

- **consumeThreshold attribute** (*Int32*)
  Price responsive clients acting upon the associated RateComponent SHOULD consume the associated commodity while the price is less than this threshold.

- **maxReductionThreshold attribute** (*Int32*)
  Price responsive clients acting upon the associated RateComponent SHOULD reduce consumption to the maximum extent possible while the price is greater than this threshold.

**PriceResponseCfgList Object** (List)
A List element to hold PriceResponseCfg objects.

**TimeConfiguration Object** ()
Contains attributes related to the configuration of the time service.

- **dstEndRule attribute** (*DstRuleType*)
  Rule to calculate end of daylight savings time in the current year. Result of dstEndRule must be greater than result of dstStartRule.

- **dstOffset attribute** (*TimeOffsetType*)
  Daylight savings time offset from local standard time.

- **dstStartRule attribute** (*DstRuleType*)
  Rule to calculate start of daylight savings time in the current year. Result of dstEndRule must be greater than result of dstStartRule.

- **tzOffset attribute** (*TimeOffsetType*)
  Local time zone offset from UTCTime. Does not include any daylight savings time offsets.
15.1.13 SoftwareDownload Package

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

*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.

*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

*IFDI attribute* (HexBinary160) [0..1]

This element MUST be set to the LFDI of the device for which this file is targeted.

*mfHwVer attribute* (String32) [0..1]

This element MUST be set to the hardware version for which this file is targeted.

*mfID attribute* (PENType)

This element MUST be set to the manufacturer's Private Enterprise Number (assigned by IANA).
mfModel attribute (String32)
This element MUST be set to the manufacturer model number for which this file is targeted. The syntax
and semantics are left to the manufacturer.

mfSerNum attribute (String32) [0..1]
This element MUST be set to the manufacturer serial number for which this file is targeted. The syntax
and semantics are left to the manufacturer.

mfVer attribute (String16)
This element MUST be set to the software version information for this file. The syntax and semantics are
left to the manufacturer.

size attribute (UInt32)
This element MUST be set to the total size (in bytes) of the file referenced by fileURI.

type attribute (HexBinary16)
A value indicating the type of the file. MUST be one of the following values:
 00 = Software Image
 01 = Security Credential
 02 = Configuration
 03 = Log
 04–7FFF = SEP2 reserved
 8000–FFFF = Manufacturer defined

FileList Object (List)
A List element to hold File objects.

FileStatus Object (Resource)
This object provides status of device file load and activation operations.

activateTime attribute (TimeType) [0..1]
Date/time at which this File, referred to by FileLink, will be activated. Omission of or presence and value
of this element MUST exactly match omission or presence and value of the activateTime element from
the File resource.

loadPercent attribute (UInt8)
This element MUST be set to the percentage of the file, indicated by FileLink, that was loaded during the
latest load attempt. This value MUST be reset to 0 each time a load attempt is started for the File
indicated by FileLink. This value MUST be increased when an LD receives HTTP response containing
file content. This value MUST be set to 100 when the full content of the file has been received by the LD

nextRequestAttempt attribute (TimeType)
This element MUST be set to the time at which the LD will issue its next GET request for file content
from the File indicated by FileLink

request503Count attribute (UInt16)
This value MUST be reset to 0 when FileLink is first pointed at a new File. This value MUST be
incremented each time an
LD receives a 503 error from the FS.
**requestFailCount attribute (UInt16)**

This value MUST be reset to 0 when FileLink is first pointed at a new File. This value MUST be incremented each time a GET request for file content failed. 503 errors MUST be excluded from this counter.

**status attribute (UInt8)**

Current loading status of the file indicated by FileLink. This element MUST be set to one of the following values:

- 0 - No load operation in progress
- 1 - File load in progress (first request for file content has been issued by LD)
- 2 - File load failed
- 3 - File loaded successfully (full content of file has been received by the LD), signature verification in progress
- 4 - File signature verification failed
- 5 - File signature verified, waiting to activate file.
- 6 - File activation failed
- 7 - File activation in progress
- 8 - File activated successfully (this state may not be reached/persisted through an image activation)
- 9-255 - Reserved for future use.

**statusTime attribute (TimeType)**

This element MUST be set to the time at which file status transitioned to the value indicated in the status element.
15.1.14 **DRLC Package**

Contains definitions for Demand Response Load Control functionality.

![DRLC Event Diagram]

---

**Figure 15-21: DRLC Event**
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.

type attribute (ApplianceLoadReductionType)
Indicates the type of appliance load reduction requested.

DemandResponseProgram Object (IdentifiedObject)
Demand response program.

availabilityUpdatePercentChangeThreshold attribute (PerCent) [0..1]
This attribute allows program providers to specify the requested granularity of updates to LoadShedAvailability sheddablePercent. 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 sheddablePercent is greater than availabilityUpdatePercentChangeThreshold.

availabilityUpdatePowerChangeThreshold attribute (ActivePower) [0..1]
This attribute allows program providers to specify the requested granularity of updates to LoadShedAvailability sheddablePower. 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 sheddablePower is greater than availabilityUpdatePowerChangeThreshold.
primacy attribute (PrimacyType)
Indicates the relative primacy of the provider of this program.

DemandResponseProgramList Object (SubscribableList)
A List element to hold DemandResponseProgram objects.

DutyCycle Object ()
Duty cycle control is a device specific issue and is managed by the device. The duty cycle of the
device under control should span the shortest practical time period in accordance with the nature
of the device under control and the intent of the request for demand reduction. The default
factory setting SHOULD be three minutes for each 10% of duty cycle. This indicates that the
default time period over which a duty cycle is applied is 30 minutes, meaning a 10% duty cycle
would cause a device to be ON for 3 minutes. The “off state” SHALL precede the “on state”.

normalValue attribute (UInt8)
Contains the maximum On state duty cycle applied by the end device, as a percentage of time. The field
not present indicates that this field has not been used by the end device.

EndDeviceControl Object (RandomizableEvent)
Instructs an EndDevice to perform a specified action.

deviceCategory attribute (DeviceCategoryType)
Specifies the bitmap indicating the categories of devices that SHOULD respond. Devices SHOULD
ignore events that do not indicate their device category.

drProgramMandatory attribute (boolean)
A flag to indicate if the EndDeviceControl is considered a mandatory event as defined by the service
provider issuing the EndDeviceControl. The drProgramMandatory flag alerts the client/user that they will
be subject to penalty or ineligibility based on the service provider’s program rules for that
EndDeviceCategory.

loadShiftForward attribute (boolean)
Indicates that the event intends to increase consumption. A value of true indicates the intention to increase
usage value, and a value of false indicates the intention to decrease usage.

overrideDuration attribute (UInt16) [0..1]
The overrideDuration attribute provides a duration, in seconds, for which a client device is allowed to
override this EndDeviceControl and still meet the contractual agreement with a service provider without
opting out. If overrideDuration is not specified, then it SHALL default to 0.

EndDeviceControlList Object (SubscribableList)
A List element to hold EndDeviceControl objects.

LoadShedAvailability Object (Resource)
Indicates current consumption status and ability to shed load.

availabilityDuration attribute (UInt32) [0..1]
Indicates for how many seconds the consuming device will be able to reduce consumption at the
maximum response level.

sheddablePercent attribute (PerCent) [0..1]
Maximum percent of current operating load that is estimated to be sheddable.

sheddablePower attribute (ActivePower) [0..1]
Maximum amount of current operating load that is estimated to be sheddable, in Watts.

**Offset Object ()**

If a temperature offset is sent that causes the heating or cooling temperature set point to exceed the limit boundaries that are programmed into the device, the device SHALL respond by setting the temperature at the limit.

If an EDC is being targeted at multiple devices or to a device that controls multiple devices (e.g., EMS), it can provide multiple Offset types within one EDC. For events with multiple Offset types, a client SHALL select the Offset that best fits their operating function.

Alternatively, an event with a single Offset type can be targeted at an EMS in order to request a percentage load reduction on the average energy usage of the entire premise. An EMS SHOULD use the Metering function set to determine the initial load in the premise, reduce energy consumption by controlling devices at its disposal, and at the conclusion of the event, once again use the Metering function set to determine if the desired load reduction was achieved.

**coolingOffset attribute (UInt8) [0..1]**

The value change requested for the cooling offset, in degree C / 10. The value should be added to the normal set point for cooling, or if loadShiftForward is true, then the value should be subtracted from the normal set point.

**heatingOffset attribute (UInt8) [0..1]**

The value change requested for the heating offset, in degree C / 10. The value should be subtracted for heating, or if loadShiftForward is true, then the value should be added to the normal set point.

**loadAdjustmentPercentageOffset attribute (UInt8) [0..1]**

The value change requested for the load adjustment percentage, in tenths of a percent. The value should be subtracted from the normal setting, or if loadShiftForward is true, then the value should be added to the normal setting.

**SetPoint Object ()**

The SetPoint object is used to apply specific temperature set points to a temperature control device. The values of the heatingSetpoint and coolingSetpoint attributes SHALL be calculated as follows:

Cooling/Heating Temperature Set Point / 100 = temperature in degrees Celsius where -273.15°C <= temperature <= 327.67°C, corresponding to a Cooling and/or Heating Temperature Set Point. The maximum resolution this format allows is 0.01°C.

The field not present in a Response indicates that this field has not been used by the end device.

If a temperature is sent that exceeds the temperature limit boundaries that are programmed into the device, the device SHALL respond by setting the temperature at the limit.

**coolingSetpoint attribute (Int16) [0..1]**

This attribute represents the cooling temperature set point in degrees Celsius / 100. (Hundredths of a degree C)

**heatingSetpoint attribute (Int16) [0..1]**

This attribute represents the heating temperature set point in degrees Celsius / 100. (Hundredths of a degree C)
TargetReduction Object

The TargetReduction object is used by a Demand Response service provider to provide a RECOMMENDED threshold that a device/premises should maintain its consumption below. For example, a service provider can provide a RECOMMENDED threshold of some kWh for a 3-hour event. This means that the device/premises would maintain its consumption below the specified limit for the specified period.

**type attribute** (UnitType)

Indicates the type of reduction requested.

**value attribute** (UInt16)

Indicates the requested amount of the relevant commodity to be reduced.

15.1.15 Metering Package

Contains definitions related to measurements of energy at usage points.
Metering Data Types

Data types based on CIM 61968-9 Annex C, D 2010-12-06 version

<table>
<thead>
<tr>
<th>PhaseCode</th>
<th>PowerOfMultipleType</th>
<th>AccumulationBehaviourType</th>
<th>KindType</th>
<th>UomType</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FlowDirectionType</th>
<th>DataQualifierType</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ConsumptionBlockType</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Figure 15-24: Metering Data Types

- **MeterReading Object** (MeterReadingBase)
  - Set of values obtained from the meter.
- **MeterReadingList Object** (SubscriberableList)
  - 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.

  **subscriberable attribute** (SubscriberableType) [0..1] «XSDAttribute»
  - 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** (SubscriberableList)
  - A list element to hold Reading objects.
- **ReadingSet Object** (ReadingSetBase)
  - A set of Readings of the ReadingType indicated by the parent MeterReading.
- **ReadingSetList Object** (SubscriberableList)
  - 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.
The “accumulation behaviour” indicates how the value is represented to accumulate over time.

The amount of heat generated when a given mass of fuel is completely burned. The CalorificValue is used to convert the measured volume or mass of gas into kWh. The CalorificValue attribute represents the current active value.

The commodity applicable to this ReadingType.

Accounts for changes in the volume of gas based on temperature and pressure. The ConversionFactor attribute represents the current active value. The ConversionFactor is dimensionless. The default value for the ConversionFactor is 1, which means no conversion is applied. A price server can advertise a new/different value at any time.

The data type can be used to describe a salient attribute of the data. Possible values are average, absolute, and etc.

Anything involving current might have a flow direction. Possible values include forward and reverse.

Default interval length specified in seconds.

Compound class that contains kindCategory and kindIndex

To be populated for mirrors of interval data to set the expected number of intervals per ReadingSet. Servers may discard intervals received that exceed this number.

Number of consumption blocks. 0 means not applicable, and is the default if not specified. The value needs to be at least 1 if any actual prices are provided.

The number of TOU tiers that can be used by any resource configured by this ReadingType. Servers SHALL populate this value with the largest touTier value that will ever be used while this ReadingType is in effect. Servers SHALL set numberOfTouTiers equal to the number of standard TOU tiers plus the number of CPP tiers that may be used while this ReadingType is in effect. Servers SHALL specify a value between 1 and 255 (inclusive) for numberOfTouTiers (servers providing flat rate pricing SHALL set numberOfTouTiers to 1, as in practice there is no difference between having no tiers and having one tier).

Contains phase information associated with the type.

Indicates the power of ten multiplier applicable to the unit of measure of this ReadingType.
subIntervalLength attribute (UInt32) [0..1]
Default sub-interval length specified in seconds for Readings of ReadingType. Some demand calculations
are done over a number of smaller intervals. For example, in a rolling demand calculation, the demand
value is defined as the rolling sum of smaller intervals over the intervalLength. The subintervalLength is
the length of the smaller interval in this calculation. It SHALL be an integral division of the
intervalLength. The number of sub-intervals can be calculated by dividing the intervalLength by the
subintervalLength.

supplyLimit attribute (UInt48) [0..1]
Reflects the supply limit set in the meter. This value can be compared to the Reading value to understand
if limits are being approached or exceeded. Units follow the same definition as in this ReadingType.

tieredConsumptionBlocks attribute (boolean) [0..1]
Specifies whether or not the consumption blocks are differentiated by TOUTier or not. Default is false, if
not specified.
true = consumption accumulated over individual tiers
false = consumption accumulated over all tiers

tieredConsumptionBlocks attribute (boolean) [0..1]
Indicates the measurement type for the units of measure for the readings of this type.

UsagePoint Object (UsagePointBase)
Logical point on a network at which consumption or production is either physically measured
(e.g. metered) or estimated (e.g. unmetered street lights).

UsagePointList Object (SubscribableList)
A List element to hold UsagePoint objects.
15.1.15.1 **Metering Mirror Package**

**Figure 15-25: Metering Mirror**
Figure 15-26: Metering Mirror Inheritance

**MirrorMeterReading Object** (MeterReadingBase)
Mimic of MeterReading used for managing mirrors.

- **lastUpdateTime attribute** (TimeType) [0..1]
The date and time of the last update.

- **nextUpdateTime attribute** (TimeType) [0..1]
The date and time of the next planned update.

**MirrorMeterReadingList Object** (List)
A List of MirrorMeterReading instances.

**MeterReadingBase Object** (IdentifiedObject)
A container for associating ReadingType, Readings and ReadingSets.

**MirrorReadingSet Object** (ReadingSetBase)
A set of Readings of the ReadingType indicated by the parent MeterReading.

**MirrorUsagePoint Object** (UsagePointBase)
A parallel to UsagePoint to support mirroring

- **deviceLFDI attribute** (HexBinary160)
The LFDI of the device being mirrored.

**MirrorUsagePointList Object**  (List)
A List of MirrorUsagePoint instances.

**ReadingBase Object**  (Resource)
Specific value measured by a meter or other asset. ReadingBase is abstract, used to define the elements common to Reading and IntervalReading.

- **consumptionBlock attribute** (*ConsumptionBlockType*)  [0..1]
  Indicates the consumption block related to the reading. REQUIRED if ReadingType numberOfConsumptionBlocks is non-zero. If not specified, is assumed to be "0 - N/A".

- **qualityFlags attribute** (*HexBinary16*)  [0..1]
  List of codes indicating the quality of the reading, using specification:
  - Bit 0 - valid: data that has gone through all required validation checks and either passed them all or has been verified
  - Bit 1 - manually edited: Replaced or approved by a human
  - Bit 2 - estimated using reference day: data value was replaced by a machine computed value based on analysis of historical data using the same type of measurement.
  - Bit 3 - estimated using linear interpolation: data value was computed using linear interpolation based on the readings before and after it
  - Bit 4 - questionable: data that has failed one or more checks
  - Bit 5 - derived: data that has been calculated (using logic or mathematical operations), not necessarily measured directly
  - Bit 6 - projected (forecast): data that has been calculated as a projection or forecast of future readings

- **timePeriod attribute** (*DateTimeInterval*)  [0..1]
  The time interval associated with the reading. If not specified, then defaults to the intervalLength specified in the associated ReadingType.

- **touTier attribute** (*TOUType*)  [0..1]
  Indicates the time of use tier related to the reading. REQUIRED if ReadingType numberOfTouTiers is non-zero. If not specified, is assumed to be "0 - N/A".

- **value attribute** (*Int48*)  [0..1]
  Value in units specified by ReadingType

**ReadingSetBase Object**  (IdentifiedObject)
A set of Readings of the ReadingType indicated by the parent MeterReading. ReadingBase is abstract, used to define the elements common to ReadingSet and IntervalBlock.

- **timePeriod attribute** (*DateTimeInterval*)
  Specifies the time range during which the contained readings were taken.

**UsagePointBase Object**  (IdentifiedObject)
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 ReadingType, Readings and ReadingSets.
roleFlags attribute (RoleFlagsType)
Specifies the roles that apply to the usage point.

serviceCategoryKind attribute (ServiceKind)
The kind of service provided by this usage point.

status attribute (UInt8)
Specifies the current status of the service at this usage point.

0 = off
1 = on

15.1.16 Pricing Package
Contains definitions of information related to price.

Figure 15-27: Pricing

ConsumptionTariffInterval Object (Resource)
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
simultaneously defines the entry value of this step and the closing value of the previous step.
Where consumption is greater than startValue, it falls within this block and where consumption is
less than or equal to startValue, it falls within one of the previous blocks.

**consumptionBlock attribute** (ConsumptionBlockType)
Indicates the consumption block related to the reading. If not specified, is assumed to be "0 - N/A".

**price attribute** (Int32) [0..1]
The charge for this rate component, per unit of measure defined by the associated ReadingType, in
currency specified in TariffProfile.
The Pricing service provider determines the appropriate price attribute value based on its applicable
regulatory rules. For example, price could be net or inclusive of applicable taxes, fees, or levies.
The Billing function set provides the ability to represent billing information in a more detailed manner.

**startValue attribute** (UInt48)
The lowest level of consumption that defines the starting point of this consumption step or block.
Thresholds start at zero for each billing period.
If specified, the first ConsumptionTariffInterval.startValue for a TimeTariffInterval instance SHALL begin
at "0." Subsequent ConsumptionTariffInterval.startValue elements SHALL be greater than the previous
one.

**ConsumptionTariffIntervalList Object** (List)
A List element to hold ConsumptionTariffInterval objects.

**CostKindType Object** (UInt8)
0 - Carbon Dioxide emissions, in grams per unit
1 - Sulfur Dioxide emissions, in grams per unit
2 - Nitrogen Oxides emissions, in grams per unit
3 - Renewable generation, as a percentage of overall generation
All other values reserved.

**EnvironmentalCost Object** ()
Provides alternative or secondary price information for the relevant RateComponent. Supports
jurisdictions that seek to convey the environmental price per unit of the specified commodity not
expressed in currency.
Implementers and consumers can use this attribute to prioritize operations of their HAN devices
(e.g., PEV charging during times of high availability of renewable electricity resources).

**amount attribute** (UInt32)
The estimated or actual environmental or other cost, per commodity unit defined by the ReadingType, for
this RateComponent (e.g., grams of carbon dioxide emissions each per kWh).

**costKind attribute** (CostKindType)
The kind of cost referred to in the amount.

**costLevel attribute** (UInt8)
The relative level of the amount attribute. In conjunction with numCostLevels, this attribute informs a device of the relative scarcity of the amount attribute (e.g., a high or low availability of renewable generation).

numCostLevels and costLevel values SHALL ascend in order of scarcity, where "0" signals the lowest relative cost and higher values signal increasing cost. For example, if numCostLevels is equal to "3," then if the lowest relative costLevel were equal to "0," devices would assume this is the lowest relative period to operate. Likewise, if the costLevel in the next TimeTariffInterval instance is equal to "1," then the device would assume it is relatively more expensive, in environmental terms, to operate during this TimeTariffInterval instance than the previous one.

There is no limit to the number of relative price levels other than that indicated in the attribute type, but for practicality, service providers should strive for simplicity and recognize the diminishing returns derived from increasing the numCostLevel value greater than four.

**numCostLevels attribute** (UInt8)

The number of all relative cost levels.

In conjunction with costLevel, numCostLevels signals the relative scarcity of the commodity for the duration of the TimeTariffInterval instance (e.g., a relative indication of cost). This is useful in providing context for nominal cost signals to consumers or devices that might see a range of amount values from different service providers or from the same service provider.

**RateComponent Object** (IdentifiedObject)

Specifies the applicable charges for a single component of the rate, which could be generation price or consumption price, for example.

**flowRateEndLimit attribute** (UnitValueType) [0..1]

Specifies the maximum flow rate (e.g., kW for electricity) for which this RateComponent applies, for the usage point and given rate / tariff.

In combination with flowRateStartLimit, allows a service provider to define the demand or output characteristics for the particular tariff design. If a server includes the flowRateEndLimit attribute, then it SHALL also include flowRateStartLimit attribute.

For example, a service provider’s tariff limits customers to 20 kWs of demand for the given rate structure. Above this threshold (from 20-50 kWs), there are different demand charges per unit of consumption. The service provider can use flowRateStartLimit and flowRateEndLimit to describe the demand characteristics of the different rates. Similarly, these attributes can be used to describe limits on premises DERs that might be producing a commodity and sending it back into the distribution network.

Note: At the time of writing, service provider tariffs with demand-based components were not originally identified as being in scope, and service provider tariffs vary widely in their use of demand components and the method for computing charges. It is expected that industry groups (e.g., OpenSG) will document requirements in the future that the SEP 2.0 community can then use as source material for the next version of SEP 2.0.

**flowRateStartLimit attribute** (UnitValueType) [0..1]

Specifies the minimum flow rate (e.g., kW for electricity) for which this RateComponent applies, for the usage point and given rate / tariff.

In combination with flowRateEndLimit, allows a service provider to define the demand or output characteristics for the particular tariff design. If a server includes the flowRateStartLimit attribute, then it
SHALL also include flowRateEndLimit attribute.

**roleFlags attribute** (*RoleFlagsType*)

Specifies the roles that this usage point has been assigned.

**RateComponentList Object** (*List*)

A List element to hold RateComponent objects.

**TariffProfile Object** (*IdentifiedObject*)

A schedule of charges; structure that allows the definition of tariff structures such as step (block) and time of use (tier) when used in conjunction with TimeTariffInterval and ConsumptionTariffInterval.

**currency attribute** (*CurrencyCode*) [0..1]

The currency code indicating the currency for this TariffProfile.

**pricePowerOfTenMultiplier attribute** (*PowerOfTenMultiplierType*) [0..1]

Indicates the power of ten multiplier for the price attribute.

**primacy attribute** (*PrimacyType*)

Indicates the relative primacy of the provider of this program.

**rateCode attribute** (*String20*) [0..1]

The rate code for this tariff profile. Provided by the Pricing service provider per its internal business needs and practices and provides a method to identify the specific rate code for the TariffProfile instance. This would typically not be communicated to the user except to facilitate troubleshooting due to its service provider-specific technical nature.

**serviceCategoryKind attribute** (*ServiceKind*)

The kind of service provided by this usage point.

**TariffProfileList Object** (*SubscribableList*)

A List element to hold TariffProfile objects.

**TimeTariffInterval Object** (*RandomizableEvent*)

Describes the time-differentiated portion of the RateComponent, if applicable, and provides the ability to specify multiple time intervals, each with its own consumption-based components and other attributes.

**touTier attribute** (*TOUType*)

Indicates the time of use tier related to the reading. If not specified, is assumed to be "0 - N/A".

**TimeTariffIntervalList Object** (*SubscribableList*)

A List element to hold TimeTariffInterval objects.
15.1.17 **Messaging Package**

Contains text message definitions.

![Diagram of TextMessage class and associated objects]

**MessagingProgram Object** (SubscribableIdentifiedObject)

 Provides a container for collections of text messages.

 *locale attribute* *(LocaleType)*

 Indicates the language and region of the messages in this collection.

 *primacy attribute* *(PrimacyType)*

 Indicates the relative primacy of the provider of this program.
**MessagingProgramList Object** (SubscribableList)
A List element to hold MessagingProgram objects.

**PriorityType Object** (UInt8)
Indicates the priority of a message:

0 - Low
1 - Normal
2 - High
3 - Critical
All other values reserved.

**TextMessage Object** (Event)
Text message such as a notification.

- **originator attribute** (String20) [0..1]
  Indicates the human-readable name of the publisher of the message
- **priority attribute** (PriorityType)
The priority is used to inform the client of the priority of the particular message. Devices with constrained or limited resources for displaying Messages should use this attribute to determine how to handle displaying currently active Messages (e.g. if a device uses a scrolling method with a single Message viewable at a time it MAY want to push a low priority Message to the background and bring a newly received higher priority Message to the foreground).
- **textMessage attribute** (string)
The textMessage attribute contains the actual UTF-8 encoded text to be displayed in conjunction with the messageLength attribute which contains the overall length of the textMessage attribute. Clients and servers SHALL support a reception of a Message of 100 bytes in length. Messages that exceed the clients display size will be left to the client to choose what method to handle the message (truncation, scrolling, etc.).

**TextMessageList Object** (SubscribableList)
A List element to hold TextMessage objects.
15.1.18 **Billing Package**

Contains representations of charges and other billing related information.

![Diagram of Billing Package](image_url)

**Figure 15-29: Billing**
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.

**billLastPeriod attribute** (*Int48*) [0..1]

The amount of the bill for the previous billing period.

**billToDate attribute** (*Int48*) [0..1]

The bill amount related to the billing period as of the statusTimeStamp.

**interval attribute** (*DateTimeInterval*)

The time interval for this billing period.

**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.
BillingMeterReadingBase Object (MeterReadingBase)
Contains historical, target, and projection readings of various types, possibly associated with charges.

BillingReading Object (ReadingBase)
Data captured at regular intervals of time. Interval data could be captured as incremental data, absolute data, or relative data. The source for the data is usually a tariff quantity or an engineering quantity. Data is typically captured in time-tagged, uniform, fixed-length intervals of 5 min, 10 min, 15 min, 30 min, or 60 min. However, consumption aggregations can also be represented with this class.

BillingReadingList Object (List)
A List element to hold BillingReading objects.

BillingReadingSet Object (ReadingSetBase)
Time sequence of readings of the same reading type.

BillingReadingSetList Object (SubscribableList)
A List element to hold BillingReadingSet objects.

Charge Object ()
Charges contain charges on a customer bill. These could be items like taxes, levies, surcharges, rebates, or others. This is meant to allow the HAN device to retrieve enough information to be able to reconstruct an estimate of what the total bill would look like.

Providers can provide line item billing, including multiple charge kinds (e.g. taxes, surcharges) at whatever granularity desired, using as many Charges as desired during a billing period. There can also be any number of Charges associated with different ReadingTypes to distinguish between TOU tiers, consumption blocks, or demand charges.

description attribute (String20) [0..1]
A description of the charge.

kind attribute (ChargeKind) [0..1]
The type (kind) of charge.

value attribute (Int32)
A monetary charge.

ChargeKind Object (UInt8)
Kind of charge.

0 - Consumption Charge
1 - Rebate
2 - Auxiliary Charge
3 - Demand Charge
4 - Tax Charge

CustomerAccount Object (IdentifiedObject)
Assignment of a group of products and services purchased by the Customer through a CustomerAgreement, used as a mechanism for customer billing and payment. It contains common information from the various types of CustomerAgreements to create billings (invoices) for a
Customer and receive payment.

**currency attribute** (UInt16)

The ISO 4217 code indicating the currency applicable to the bill amounts in the summary. See list at http://www.unece.org/cefact/recommendations/rec09/rec09_ecetrd203.pdf

**customerAccount attribute** (String42) [0..1]

The account number for the customer (if applicable).

**customerName attribute** (String42) [0..1]

The name of the customer.

**pricePowerOfTenMultiplier attribute** (PowerOfTenMultiplierType)

Indicates the power of ten multiplier for the prices in this function set.

**CustomerAccountList Object** (SubscribableList)

A List element to hold CustomerAccount objects.

**CustomerAgreement Object** (IdentifiedObject)

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]

The account number of the service account (if applicable).

**serviceLocation attribute** (String42) [0..1]

The address or textual description of the service location.

**CustomerAgreementList Object** (SubscribableList)

A List element to hold CustomerAgreement objects.

**HistoricalReading Object** (BillingMeterReadingBase)

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)

A List element to hold HistoricalReading objects.

**ProjectionReading Object** (BillingMeterReadingBase)

Contains values that forecast a future reading for the time or interval specified.

**ProjectionReadingList Object** (List)

A List element to hold ProjectionReading objects.

**TargetReading Object** (BillingMeterReadingBase)

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)

A List element to hold TargetReading objects.

**ServiceSupplier Object** (IdentifiedObject)

Organisation that provides services to Customers.
email attribute (String32) [0..1]
E-mail address for this service supplier.

phone attribute (String20) [0..1]
Human-readable phone number for this service supplier.

providerID attribute (UInt32) [0..1]
Contains the IANA PEN for the commodity provider.

web attribute (String42) [0..1]
Website URI address for this service supplier.

ServiceSupplierList Object (List)
A List element to hold ServiceSupplier objects.
15.1.19 **Prepayment Package**

Contains definitions related to storing and using payments.

![Diagram of Prepayment Package]

**Figure 15-31: Prepayment**
AccountBalance Object (Resource)

AccountBalance contains the regular credit and emergency credit balance for this given service or commodity prepay instance. It may also contain status information concerning the balance data.

availableCredit attribute (AccountingUnit)

AvailableCredit shows the balance of the sum of credits minus the sum of charges. In a Central Wallet mode this value may be passed down to the Prepayment server via an out-of-band mechanism. In Local or ESI modes, this value may be calculated based upon summation of CreditRegister transactions minus consumption charges calculated using Metering (and possibly Pricing) function set data. This value may be negative; for instance, if disconnection is prevented due to a Supply Interruption Override.

creditStatus attribute (CreditStatusType) [0..1]

CreditStatus identifies whether the present value of availableCredit is considered OK, low, exhausted, or negative.

equipmentCredit attribute (AccountingUnit) [0..1]

EmergencyCredit is the amount of credit still available for the given service or commodity prepayment instance. If both availableCredit and emergencyCredit are exhausted, then service will typically be disconnected.

equipmentCreditStatus attribute (CreditStatusType) [0..1]

EmergencyCreditStatus identifies whether the present value of emergencyCredit is considered OK, low, exhausted, or negative.

AccountingUnit Object ()

Unit for accounting; use either 'energyUnit' or 'currencyUnit' to specify the unit for 'value'.

energyUnit attribute (RealEnergy) [0..1]

Unit of service.

monetaryUnit attribute (CurrencyCode)

Unit of currency.

multiplier attribute (PowerOfTenMultiplierType)

Multiplier for the 'energyUnit' or 'monetaryUnit'.

value attribute (Int32)

Value of the monetary aspect

CreditRegister Object (IdentifiedObject)

CreditRegister instances define a credit-modifying transaction. Typically this would be a credit-adding transaction, but may be a subtracting transaction (perhaps in response to an out-of-band debt signal).

creditAmount attribute (AccountingUnit)

CreditAmount is the amount of credit being added by a particular CreditRegister transaction. Negative values indicate that credit is being subtracted.

creditType attribute (CreditTypeType) [0..1]

CreditType indicates whether the credit transaction applies to regular or emergency credit.

effectiveTime attribute (TimeType)

EffectiveTime identifies the time at which the credit transaction goes into effect. For credit addition transactions, this is typically the moment at which the transaction takes place. For credit subtraction
transactions, (e.g., non-fuel debt recovery transactions initiated from a back-haul or ESI) this may be a future time at which credit is deducted.

**token attribute (String32)**

Token is security data that authenticates the legitimacy of the transaction. The details of this token are not defined by Smart Energy 2.0. How a Prepayment server handles this field is left as vendor specific implementation or will be defined by one or more other standards.

**CreditRegisterList Object** (List)

A List element to hold CreditRegister objects.

**Prepayment Object** (IdentifiedObject)

Prepayment (inherited from CIM SDPAccountingFunction)

**creditExpiryLevel attribute (AccountingUnit) [0..1]**

CreditExpiryLevel is the set point for availableCredit at which the service level may be changed. The typical value for this attribute is 0, regardless of whether the account balance is measured in a monetary or commodity basis. The units for this attribute SHALL match the units used for availableCredit.

**lowCreditWarningLevel attribute (AccountingUnit) [0..1]**

LowCreditWarningLevel is the set point for availableCredit at which the creditStatus attribute in the AccountBalance resource SHALL indicate that available credit is low. The units for this attribute SHALL match the units used for availableCredit. Typically, this value is set by the service provider.

**lowEmergencyCreditWarningLevel attribute (AccountingUnit) [0..1]**

LowEmergencyCreditWarningLevel is the set point for emergencyCredit at which the creditStatus attribute in the AccountBalance resource SHALL indicate that emergencycredit is low. The units for this attribute SHALL match the units used for availableCredit. Typically, this value is set by the service provider.

**prepayMode attribute (PrepayModeType)**

PrepayMode specifies whether the given Prepayment instance is operating in Credit, Central Wallet, ESI, or Local prepayment mode. The Credit mode indicates that prepayment is not presently in effect. The other modes are described in the Overview Section above.

**PrepaymentList Object** (SubscribableList)

A List element to hold Prepayment objects.

**PrepayModeType Object** (UInt8)

0 - Central Wallet
1 - ESI
2 - Local
3 - Credit
All other values reserved.

**PrepayOperationStatus Object** (Resource)

PrepayOperationStatus describes the status of the service or commodity being conditionally controlled by the Prepayment function set.

**creditTypeChange attribute (CreditTypeChange) [0..1]**

CreditTypeChange is used to define a pending change of creditTypeInUse, which will activate at a
specified time.

**creditTypeInUse attribute** *(CreditTypeType) [0..1]*

CreditTypeInUse identifies whether the present mode of operation is consuming regular credit or emergency credit.

**serviceChange attribute** *(ServiceChange) [0..1]*

ServiceChange is used to define a pending change of serviceStatus, which will activate at a specified time.

**serviceStatus attribute** *(ServiceStatusType)*

ServiceStatus identifies whether the service is connected or disconnected, or armed for connection or disconnection.

**ServiceChange Object ()**

Specifies a change to the service status.

**newStatus attribute** *(ServiceStatusType)*

The new service status, to take effect at the time specified by startTime

**startTime attribute** *(TimeType)*

The date/time when the change is to take effect.

**SupplyInterruptionOverride Object** *(Resource)*

SupplyInterruptionOverride: There may be periods of time when social, regulatory or other concerns mean that service should not be interrupted, even when available credit has been exhausted. Each Prepayment instance links to a List of SupplyInterruptionOverride instances.

Each SupplyInterruptionOverride defines a contiguous period of time during which supply SHALL NOT be interrupted.

**description attribute** *(String32) [0..1]*

The description is a human readable text describing or naming the object.

**interval attribute** *(DateTimeInterval)*

Interval defines the period of time during which supply should not be interrupted.

**SupplyInterruptionOverrideList Object** *(List)*

A List element to hold SupplyInterruptionOverride objects.

**CreditStatusType Object** *(UInt8)*

0 - Credit Ok
1 - Credit Low
2 - Credit Exhausted
3 - Credit Negative
All other values reserved.

**CreditTypeType Object** *(UInt8)*

0 - Regular
1 - Emergency
2 - Regular, then Emergency
3 - Emergency, then Regular

All other values reserved.

CreditTypeChange Object ()

Specifies a change to the credit type.

newType attribute (CreditTypeType)

The new credit type, to take effect at the time specified by startTime

startTime attribute (TimeType)

The date/time when the change is to take effect.

ServiceStatusType Object (UInt8)

0 - Connected

1 - Disconnected

2 - Armed for Connect

3 - Armed for Disconnect

4 - No Contactor

5 - Load Limited

All other values reserved.
15.1.20 **FlowReservation Package**

Contains flow (charge) reservation model to allow fine-grained control of high-demand loads such as fast-charging batteries.

---

**FlowReservation Request**

```
AbstractDevice
EndDevice

FlowReservationRequestListLink

List

FlowReservationRequestList

FlowReservationRequest

+ creationTime :TimeType
+ durationRequested :UInt16 [0..1]
+ energyRequested :SignedRealEnergy
+ intervalRequested :DateTimeInterval
+ powerRequested :ActivePower
+ mRID :mRIDType
+ version :VersionType [0..1] = 0
<br>«XSDattribute»
+ description :String32 [0..1]
+ href :anyURI [0..1]
```

**FlowReservation Response**

```
FlowReservationResponseListLink

FlowReservationResponseList

FlowReservationResponse

+ energyAvailable :RealEnergy
+ powerAvailable :ActivePower
+ subject :mRIDType
+ creationTime :TimeType
+ interval :DateTimeInterval
+ mRID :mRIDType
+ version :VersionType [0..1] = 0
<br>«XSDattribute»
+ description :String32 [0..1]
+ href :anyURI [0..1]
+ replyTo :anyURI [0..1]
+ responseRequired :HexBinary8 [0..1] = 00
<br>«Resource»
```

**RequestStatus Object**

The RequestStatus object is used to indicate the current status of a Flow Reservation Request.

```
dateTime attribute (TimeType)
```

---

Notes:
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.
The dateTime attribute will provide a timestamp of when the request status was set. dateTime MUST be set to the time at which the status change occurred, not a time in the future or past.

**requestStatus attribute (UInt8)**
Field representing the request status type.

0 = Requested
1 = Cancelled
All other values reserved.

**AbstractFlowReservation Object** (Event)
Provides definition of FlowReservation elements in common between Requests and Responses.

**FlowReservationRequest Object** (IdentifiedObject)
Used to request flow transactions. Client EndDevices submit a request for charging or discharging from the server. The server creates an associated FlowReservationResponse containing the charging parameters and interval to provide a lower aggregated demand at the premises, or within a larger part of the distribution system.

**creationTime attribute (TimeType)**
The time at which the request was created.

**durationRequested attribute (UInt16) [0..1]**
A value that is calculated by the storage device that defines the minimum duration, in seconds, that it will take to complete the actual flow transaction, including any ramp times and conditioning times, if applicable.

**energyRequested attribute (SignedRealEnergy)**
Indicates the total amount of energy, in Watt-Hours, requested to be transferred between the storage device and the electric power system. Positive values indicate charging and negative values indicate discharging. This sign convention is different than for the DER function where discharging is positive.
Note that the energyRequestNow attribute in the PowerStatus Object must always represent a charging solution and it is not allowed to have a negative value.

**intervalRequested attribute (DateTimeInterval)**
The time window during which the flow reservation is needed. For example, if an electric vehicle is set with a 7:00 AM time charge is needed, and price drops to the lowest tier at 11:00 PM, then this window would likely be from 11:00 PM until 7:00 AM.

**powerRequested attribute (ActivePower)**
Indicates the sustained level of power, in Watts, that is requested. For charging this is calculated by the storage device and it represents the charging system capability (which for an electric vehicle must also account for any power limitations due to the EVSE control pilot). For discharging, a lower value than the inverter capability can be used as a target.

**FlowReservationRequestList Object** (List)
A List element to hold FlowReservationRequest objects.

**FlowReservationResponse Object** (Event)
The server may modify the charging or discharging parameters and interval to provide a lower aggregated demand at the premises, or within a larger part of the distribution system.

**energyAvailable attribute (RealEnergy)**
Indicates the amount of energy available.

**powerAvailable attribute** *(ActivePower)*

Indicates the amount of power available.

**subject attribute** *(mRIDType)*

The subject field provides a method to match the response with the originating event. It is populated with the mRID of the corresponding FlowReservationRequest object.

**FlowReservationResponseList Object** *(SubscribableList)*

A List element to hold FlowReservationResponse objects.

15.1.21 **DER Package**

Contains definitions related to allowing distributed energy resources to provide energy back to the grid.

![Figure 15-33: DER Info](image-url)
### Related DER Info data types

#### DER Inverter Status Type
- **dateTime** : TimeType
- **value** : UInt8

**Notes:**
- DER Inverter Status value:
  - 0: N/A
  - 1: Off
  - 2: Sleep (auto-shutdown) or DER is at low output power/voltage
  - 3: Starting up or ON but not producing power
  - 4: Tracking MPPT power point
  - 5: Forced power reduction/derating
  - 6: Shutting down
  - 7: One or more faults exist
  - 8: Standby (service on unit) - DER may be at high output voltage/power
  - 9: Test mode
  - 10: As defined in manufacturer status
  - All other values reserved.

#### Operational Mode Status Type
- **dateTime** : TimeType
- **value** : UInt8

**Notes:**
- DER Operational Mode Status value:
  - 0: Not applicable / Unknown
  - 1: Off
  - 2: Operational mode
  - 3: Test mode
  - All other values reserved.

#### Storage Mode Status Type
- **dateTime** : TimeType
- **value** : UInt8

**Notes:**
- DER Storage Mode Status value:
  - 0: Not applicable / Unknown
  - 1: Storage charging
  - 2: Storage discharging
  - 3: Storage holding
  - All other values reserved.

#### Connect Status Type
- **dateTime** : TimeType
- **value** : UInt8

**Notes:**
- DER Connect Status value:
  - 0: N/A
  - 1: Disconnected_unavailable
  - 2: Disconnected_available
  - 3: Connected_unavailable
  - 4: Connected_available
  - 5: Connected_on
  - 6: Test mode
  - All other values reserved.

#### Local Control Mode Status Type
- **dateTime** : TimeType
- **value** : UInt8

**Notes:**
- DER Local Control Mode Status value:
  - 0: Local control
  - 1: Remote control
  - All other values reserved.

#### DER Control Type
- Notes: Control modes supported by the DER. Bit positions SHALL be defined as follows:
  - 0: Volt-VAr Mode
  - 1: Frequency-Watt Mode
  - 2: Watt-PowerFactor Mode
  - 3: Volt-Watt Mode
  - 4: Low Voltage Ride Through Mode
  - 5: High Voltage Ride Through Mode
  - 6-9: reserved
  - 10: setGenConnect
  - 11: setStorConnect
  - 12: Fixed W
  - 13: Fixed VAr
  - 14: Fixed PF
  - 15: Charge mode
  - 16: Discharge mode
  - All other values reserved.

### DER Program

**Class DER Program**

```
{1, ..}

Resource
  FunctionSetAssignmentsBase
  ListLink DERProgramListLink
  List DERProgramList

{0, ..}

IdentifiedObject DERProgram
  + primacy: PrimacyType
  :: IdentifiedObject
  : description : String32 [0..1]
  + mRID : mRIDType
  + version : VersionType [0..1] = 0
  + xSDAttributes
    + Resource
      + href : anyURI [0..1]

Device Capability
  ListLink DefaultDERControlLink
  List ActiveDERControlListLink
  List Link DERControlListLink
  ListLink DERCurveListLink

{0, ..}

IdentifiedObject PrimaryType
  + value : UInt8
    Notes: Values possible for indication of "Primary" provider:
  - 0: In home energy management system
  - 1: Contracted premises service provider
  - 2: Non-contractual service provider
  - All other values reserved.
```

### Figure 15-34: DER Info Types

![Diagram showing DER Info Types](image-url)

### Figure 15-35: DER Program

![Diagram showing DER Program](image-url)
**Figure 15-36: DER Curves**

### DER Curves

**Curvedata**
- `excitation`: Int8 [0..1]
- `xvalue`: Int32
- `yvalue`: Int32

**UNIT**

**DERCurveUnitRefType**
- `notes`
  - Specifies context for interpreting percent values:
    - 0 - N/A
    - 1 - %setMaxW
    - 2 - %setMaxVAr
    - 3 - %statVArAvail
    - 4 - %setEffectiveV
    - 5 - %setMaxChargeRate
    - 6 - %setMaxDischargeRate
    - All other values reserved.

**DERCurvetype**
- `notes`
  - 0 - Volt-Var Mode
  - 1 - Frequency-Watt Curve Mode
  - 2 - Watt-PowerFactor Mode
  - 3 - Volt-Watt Mode
  - 4 - Low Voltage Ride Through Mode
  - 5 - High Voltage Ride Through Mode
  - All other values reserved.

**PowerOfTenMultiplierType**
- `notes`
  - -9 = nano=x10^-9
  - -6 = micro=x10^-6
  - -3 = milli=x10^-3
  - 0 = none=x1 (default, if not specified)
  - 1 = deca=x10
  - 2 = hecto=x100
  - 3 = mega=x1000
  - 6 = mega=x10^6
  - 9 = Giga=x10^9
  - This is not a complete list. Any integer between -9 and 9 SHALL be supported, indicating the power of ten multiplier for the units.
**Figure 15-37: DER Control**

```
class DER Control

    IdentifiedObject
    DERProgram

    ListLink
    DERControlListLink

    SubscribableIdentifiedObject
    DefaultDERControl

    DERControlList

    DERControl

    RandomizableEvent
        + randomizeDuration :OneHourRangeType [0..1] = 0
        + randomizeStart :OneHourRangeType [0..1] = 0

    EventStatus
        + currentStatus :UInt8
        + dateTime :TimeType
        + potentiallySuperseded :boolean
        + potentiallySupersededTime :TimeType [0..1]

    ResponisbleSubscribableIdentifiedObject
    Event
        + creationTime :TimeType
        + interval :DateTimeInterval

    Link
    DefaultDERControlLink

    CurvePairType
        + lowerLimit :DERCurveLink
        + upperLimit :DERCurveLink

    DERControlBase
        + opModFixedFlow :SignedPerCent [0..1]
        + opModFixedPF :FixedPowerFactor [0..1]
        + opModFixedVAr :FixedVAr [0..1]
        + opModFixedW :PerCent [0..1]
        + opModFreqWatt :DERCurveLink [0..1]
        + opModHVRT :CurvePairType [0..1]
        + opModLVRT :CurvePairType [0..1]
        + opModVoltVAr :DERCurveLink [0..1]
        + opModVoltWatt :DERCurveLink [0..1]
        + opModWattPF :DERCurveLink [0..1]
        + rampTms :UInt16 [0..1]
```

---

Page 233  Copyright © 2013, ZigBee Alliance, Inc. and HomePlug Powerline Alliance, Inc. All rights reserved.
**DefaultDERControl Object** (SubscribableIdentifiedObject)

Contains control mode information to be used if no active DERControl is found.

**DER Object** (SubscribableResource)

Contains links to DER resources.

**DERList Object** (List)

A List element to hold DER objects.

**DERSettings Object** (SubscribableResource)

Distributed energy resource settings

- **setGenConnect attribute** (boolean) [0..1]
  
  Set generator DER as connected (true) or disconnected (false).

- **setGradW attribute** (UInt16)
  
  Set default rate of change (ramp rate) of active power output due to command or internal action, defined in %setWMax / second. Resolution is in hundredths of a percent/second and may be in the range 1 - 20000. Interpreted as a percentage change in output capability limit per second when used as a default ramp rate.

- **setMaxChargeRate attribute** (ActivePower) [0..1]
Maximum rate of energy transfer received by the storage device, in Watts. Defaults to \texttt{rtgMaxChargeRate}.

\textbf{setMaxDischargeRate attribute (ActivePower) [0..1]}

Maximum rate of energy transfer delivered by the storage device, in Watts. Defaults to \texttt{rtgMaxDischargeRate}.

\textbf{setMaxVA attribute (ApparentPower) [0..1]}

Set limit for maximum apparent power capability of the DER (in VA). Defaults to \texttt{rtgVA}.

\textbf{setMaxVAR attribute (ReactivePower) [0..1]}

Set limit for maximum reactive power delivered by the DER (in var). SHALL be a positive value <= \texttt{rtgVAR} (default).

\textbf{setMaxVARNeg attribute (ReactivePower) [0..1]}

Set limit for maximum reactive power received by the DER (in var). If present, SHALL be a negative value >= \texttt{rtgVARNeg} (default). If absent, defaults to negative \texttt{setMaxVAR}.

\textbf{setMaxW attribute (ActivePower)}

Set limit for maximum active power capability of the DER (in W). Defaults to \texttt{rtgW}.

\textbf{setMinPF attribute (UnsignedFixedPointType) [0..1]}

Set minimum Power Factor displacement limit of the DER; positive value between 0.0 (typically > 0.7) and 1.0. SHALL be >= \texttt{rtgMinPF} (default).

\textbf{setMinPFNeg attribute (FixedPointType) [0..1]}

Set minimum Power Factor displacement limit of the DER; negative value between 0.0 (typically < -0.7) and -0.9999. If present, SHALL be <= \texttt{rtgMinPFNeg} (default). If absent, defaults to negative \texttt{setMinPF}.

\textbf{setStorConnect attribute (boolean) [0..1]}

Set storage DER as connected (true) or disconnected (false).

\textbf{setVRef attribute (VoltageRMS) [0..1]}

The nominal AC voltage (RMS) at the utility's point of common coupling.

\textbf{setVRefOfs attribute (VoltageRMS) [0..1]}

The nominal AC voltage (RMS) offset between the DER's electrical connection point and the utility's point of common coupling.

\textbf{updatedTime attribute (TimeType)}

Specifies the time at which the DER information was last updated.

\textbf{DERType Object} (UInt8)

0 - Not applicable / Unknown
1 - Virtual or mixed DER
2 - Reciprocating engine
3 - Fuel cell
4 - Photovoltaic system
5 - Combined heat and power
80 - Storage (immobile)
DERAvailability Object (SubscribableResource)
Indicates current reserve generation status

- availabilityDuration attribute (UInt32) [0..1]
  Indicates number of seconds the DER will be able to deliver active power at the reservePercent level.

- maxChargeDuration attribute (UInt32) [0..1]
  Indicates number of seconds the DER will be able to receive active power at the reserveChargePercent level.

- readingTime attribute (TimeType)
  The timestamp when the DER availability was last updated.

- reserveChargePercent attribute (PerCent) [0..1]
  Percent of continuous received active power (%setMaxChargeRate) that is estimated to be available in reserve.

- reservePercent attribute (PerCent) [0..1]
  Percent of continuous delivered active power (%setMaxW) that is estimated to be available in reserve.

- statVarAvail attribute (ReactivePower) [0..1]
  Estimated reserve reactive power, in var. Represents the lesser of received or delivered reactive power.

- statWAvail attribute (ActivePower) [0..1]
  Estimated reserve active power, in watts.

DERCapability Object (Resource)
Distributed energy resource type and nameplate ratings.

- modesSupported attribute (DERControlType)
  Bitmap indicating the DER Controls implemented by the device. See DERControlType for values.

- rtgA attribute (CurrentRMS)
  Maximum continuous AC current capability of the DER, in Amperes (RMS).

- rtgAh attribute (AmpereHour) [0..1]
  Usable energy storage capacity of the DER, in AmpHours.

- rtgMaxChargeRate attribute (ActivePower) [0..1]
  Maximum rate of energy transfer received by the storage DER, in Watts.

- rtgMaxDischargeRate attribute (ActivePower) [0..1]
  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]
  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]
  Minimum Power Factor displacement capability of the DER; SHALL be a negative value between 0.0
(typically < -0.7) and -0.9999. If absent, defaults to negative \(rtg\text{MinPF}\). (Unity power factor is considered unsigned.)

\[
\textbf{rtgVA attribute (ApparentPower) } [0..1]
\]

Maximum continuous apparent power output capability of the DER, in VA.

\[
\textbf{rtgVAr attribute (ReactivePower) } [0..1]
\]

Maximum continuous reactive power delivered by the DER, in var.

\[
\textbf{rtgVArNeg attribute (ReactivePower) } [0..1]
\]

Maximum continuous reactive power received by the DER, in var. If absent, defaults to negative rtgVAr.

\[
\textbf{rtgW attribute (ActivePower)}
\]

Maximum continuous active power output capability of the DER, in watts. Represents combined generation plus storage output if DERType == 82.

\[
\textbf{rtgWh attribute (WattHour) } [0..1]
\]

Maximum energy storage capacity of the DER, in WattHours.

\[
\textbf{type attribute (DERType)}
\]

Type of DER; see DERType object

\[
\textbf{DERControlBase Object} ()
\]

Distributed Energy Resource (DER) control values.

\[
\textbf{opModFixedFlow attribute (SignedPerCent) } [0..1]
\]

The opModFixedFlow function specifies a requested charge or discharge mode setpoint, in %setMaxChargeRate if negative value or %setMaxW or %setMaxDischargeRate if positive value (in hundredths). SHALL be ignored if device is not a storage DER or setStorConnect is false.

\[
\textbf{opModFixedPF attribute (FixedPowerFactor) } [0..1]
\]

The opModFixedPF function specifies a requested fixed Power Factor (PF) setting, consisting of a signed displacement value. The PF sign (which SHALL be interpreted according to the EEI convention, where unity PF is considered unsigned) indicates the direction of reactive power flow. The actual displacement SHALL be within the limits established by setMinPF and setMinPFNeg. If issued simultaneously with other reactive power controls (e.g. opModFixedVAr) the control resulting in least var magnitude takes precedence.

\[
\textbf{opModFixedVAr attribute (FixedVAr) } [0..1]
\]

The opModFixedVAr function specifies the delivered or received reactive power limit setpoint. The context for the limit value is determined by refType and SHALL be one of %setMaxW, %setMaxVAr, or %statVArAvail. If issued simultaneously with other reactive power controls (e.g. opModFixedPF) the control resulting in least var magnitude takes precedence.

\[
\textbf{opModFixedW attribute (PerCent) } [0..1]
\]

The opModFixedW function sets the maximum active power generation level at the electrical coupling point as a percentage of set capacity (%setMaxW, in hundredths). This limitation may be met e.g. by reducing PV output or by using excess PV output to charge associated storage.

\[
\textbf{opModFreqWatt attribute (DERCurveLink) } [0..1]
\]

Specify DERCurveLink for curve type == 1. The Frequency-Watt function limits active power generation or consumption when the line frequency deviates from nominal by a specified amount. The Frequency-Watt curve is specified as an array of Frequency-Watt pairs that are interpolated into a piecewise linear function with hysteresis. The x value of each pair specifies a frequency in Hz. The y
value specifies a corresponding active power output in \(\%\text{setMaxW} \times 100\) (0 - 10000) (hundredths of a percent).

**opModHVRT attribute** *(CurvePairType)* \([0..1]\)

Specify curve pair for curve type == 5. The High Voltage Ride-Through (HVRT) function is specified by one or two duration-volt curves that define the operating region under high voltage conditions. Each HVRT curve is specified by an array of duration-volt pairs that will be interpolated into a piecewise linear function that defines an operating region. The x value of each pair specifies a duration (time at a given voltage in seconds). The y value of each pair specifies an effective percent voltage, defined as (100% * (locally measured voltage - setVRefOfs) / setVRef), in hundredths of a percent. The upperLimit curve delineates the "must disconnect" region and SHALL be defined for this control to be active. The (optional) lowerLimit curve delineates the "must remain connected" region. If the "must remain connected" curve is not specified, it is assumed to be the same as the "must disconnect" curve.

**opModLVRT attribute** *(CurvePairType)* \([0..1]\)

Specify curve pair for curve type == 4. The Low Voltage Ride-Through (LVRT) function is specified by one or two duration-volt curves that define the operating region under low voltage conditions. Each LVRT curve is specified by an array of duration-volt pairs that will be interpolated into a piecewise linear function that defines an operating region. The x value of each pair specifies a duration (time at a given voltage in seconds). The y value of each pair specifies an effective percent voltage, defined as (100% * (locally measured voltage - setVRefOfs) / setVRef), in hundredths of a percent. The lowerLimit curve delineates the "must disconnect" region and SHALL be defined for this control to be active. The (optional) upperLimit curve delineates the "must remain connected" region. If the "must remain connected" curve is not specified, it is assumed to be the same as the "must disconnect" curve.

**opModVoltVar attribute** *(DERCurveLink)* \([0..1]\)

Specify DERCurveLink for curve type == 0. The static Volt-Var function provides over- or under-excited VAr compensation as a function of measured voltage. The Volt-Var curve is specified as an array of Volt-Var pairs that are interpolated into a piecewise linear function with hysteresis. The x value of each pair specifies an effective percent voltage, defined as (100% * (locally measured voltage - setVRefOfs) / setVRef) and SHOULD support a domain of at least 0 - 13500 (in hundredths of a percent). The y value specifies a target VAr output interpreted as SignedPerCent (10000 to -10000). The meaning of the y value is determined by yRefType and must be one of %setMaxW, %setMaxVar, or %statVarAvail, all in hundredths of a percent.

**opModVoltWatt attribute** *(DERCurveLink)* \([0..1]\)

Specify DERCurveLink for curve type == 3. The Volt-Watt reduces active power output as a function of measured voltage. The Volt-Watt curve is specified as an array of Volt-Watt pairs that are interpolated into a piecewise linear function with hysteresis. The x value of each pair specifies an effective percent voltage, defined as (100% * (locally measured voltage - setVRefOfs) / setVRef) and SHOULD support a domain of at least 0 - 13500 (in hundredths of a percent). The y value specifies an active power output in %setMaxW, (0 - 10000) in hundredths of a percent.

**opModWattPF attribute** *(DERCurveLink)* \([0..1]\)

Specify DERCurveLink for curve type == 2. The Watt-PF function varies Power Factor (PF) as a function of delivered active power. The Watt-PF curve is specified as an array of Watt-PF coordinates that are interpolated into a piecewise linear function with hysteresis. The x value of each pair specifies a watt setting in %setMaxW, (0 - 10000) in hundredths of a percent. The PF output setting is a signed displacement in y value (PF sign SHALL be interpreted according to the EEI convention, where unity PF is considered unsigned). These settings are not expected to be updated very often during the life of the
installation, therefore only a single curve is required. If issued simultaneously with other reactive power controls (e.g. \texttt{opModFixedPF}) the control resulting in least var magnitude takes precedence.

\textbf{rampTms attribute (\texttt{UInt16}) [0..1]}

Requested ramp time, in hundredths of a second, for the device to transition from the current DERControl mode setting(s) to the new mode setting(s). If absent, use default ramp rate (setWGrad). Resolution is \(1/100\) sec.

\textbf{DERControl Object (RandomizableEvent)}

Distributed Energy Resource (DER) time/event-based control.

\textbf{DERControlList Object (SubscribableList)}

A List element to hold DERControl objects.

\textbf{DERControlType Object (HexBinary32)}

Control modes supported by the DER. Bit positions SHALL be defined as follows:

\begin{itemize}
\item 0 - Volt-VAr Mode
\item 1 - Frequency-Watt Mode
\item 2 - Watt-PowerFactor Mode
\item 3 - Volt-Watt Mode
\item 4 - Low Voltage Ride Through Mode
\item 5 - High Voltage Ride Through Mode
\item 6-9 - reserved
\item 10 - setGenConnect
\item 11 - setStorConnect
\item 12 - Fixed W
\item 13 - Fixed VAr
\item 14 - Fixed PF
\item 15 - Charge mode
\item 16 - Discharge mode
\end{itemize}

All other values reserved.

\textbf{DERCurve Object (IdentifiedObject)}

DER related curves such as Volt-VAr mode curves. Relationship between an independent variable (X-axis) and one or two dependent variables (Y-axis and excitation).

\textbf{creationTime attribute (\texttt{TimeType})}

The time at which the object was created.

\textbf{curveType attribute (\texttt{DERCurveType})}

Specifies the associated curve-based control mode.

\textbf{rampDecTms attribute (\texttt{UInt16}) [0..1]}

Decreasing ramp rate, interpreted as a percentage change in output capability limit per second (e.g. \texttt{\%setMaxW / sec}). Resolution is in hundredths of a percent/second and may be in the range 1 -
20000. If absent, ramp rate defaults to setWGrad.

\textbf{rampIncTms attribute (UInt16) [0..1]}
Increasing ramp rate, interpreted as a percentage change in output capability limit per second
(e.g. \%setMaxW / sec). Resolution is in hundredths of a percent/second and may be in the range 1 - 20000. If absent, ramp rate defaults to rampDecTms.

\textbf{rampPT1Tms attribute (UInt16) [0..1]}
The configuration parameter for a low-pass filter, PT1 is a time, in hundredths of a second, in which the filter will settle to 95\% of a step change in the input value. Resolution is 1/100 sec.

\textbf{xMultiplier attribute (PowerOfTenMultiplierType)}
Exponent for X-axis value.

\textbf{yMultiplier attribute (PowerOfTenMultiplierType)}
Exponent for Y-axis value.

\textbf{yRefType attribute (DERUnitRefType)}
The Y-axis units context.

\textbf{CurrentDERProgramLink Object (Link)}
SHALL contain a Link to an instance of DERProgram. If present, this is the DERProgram containing the currently active DERControl.

\textbf{DERCurveList Object (List)}
A List element to hold DERCurve objects.

\textbf{CurveData Object ()}
Data point values for defining a curve or schedule

\textbf{excitation attribute (Int8) [0..1]}
To be included with curve type 2 (Watt-PowerFactor), specifies the excitation of the power factor.
-1 = under-excited
1 = over-excited

\textbf{xvalue attribute (Int32)}
The data value of the X-axis (independent) variable, depending on the curve type. See definitions in DERControlBase for further information.

\textbf{yvalue attribute (Int32)}
The data value of the Y-axis (dependent) variable, depending on the curve type. See definitions in DERControlBase for further information.

\textbf{DERCurveType Object (UInt8)}
0 - Volt-VAr Mode
1 - Frequency-Watt Curve Mode
2 - Watt-PowerFactor Mode
3 - Volt-Watt Mode
4 - Low Voltage Ride Through Mode
5 - High Voltage Ride Through Mode
All other values reserved.

**CurvePairType Object** ()

Specifies a pair of DERCurves.

- **lowerLimit attribute** (DERCurveLink)
  - DERCurveLink for lower bound of operating region.

- **upperLimit attribute** (DERCurveLink)
  - DERCurveLink for upper bound of operating region.

**DERProgram Object** (IdentifiedObject)

Distributed Energy Resource program.

- **primacy attribute** (PrimacyType)
  - Indicates the relative primacy of the provider of this Program.

**DERProgramList Object** (List)

A List element to hold DERProgram objects.

**DERStatus Object** (SubscribableResource)

DER status information.

- **genConnectStatus attribute** (ConnectStatusType) [0..1]
  - Connect/status value for generator DER.
  - See ConnectStatusType for values.

- **inverterStatus attribute** (InverterStatusType) [0..1]
  - DER InverterStatus/value.
  - See InverterStatusType for values.

- **localControlModeStatus attribute** (LocalControlModeStatusType) [0..1]
  - The local control mode status.
  - See LocalControlModeStatusType for values.

- **manufacturerStatus attribute** (ManufacturerStatusType) [0..1]
  - Manufacturer status code.

- **operationalModeStatus attribute** (OperationalModeStatusType) [0..1]
  - Operational mode currently in use.
  - See OperationalModeStatusType for values.

- **readingTime attribute** (TimeType)
  - The timestamp when the current status was last updated.

- **stateOfChargeStatus attribute** (StateOfChargeStatusType) [0..1]
  - State of charge status.
  - See StateOfChargeStatusType for values.

- **storageModeStatus attribute** (StorageModeStatusType) [0..1]
  - Storage mode status.
  - See StorageModeStatusType for values.
**storConnectStatus attribute** *(ConnectStatusType) [0..1]*

Connect/status value for storage DER.

See ConnectStatusType for values.

**DERUnitRefType Object** *(UInt8)*

Specifies context for interpreting percent values:

0 - N/A
1 - %setMaxW
2 - %setMaxVAr
3 - %statVArAvail
4 - %setEffectiveV
5 - %setMaxChargeRate
6 - %setMaxDischargeRate
All other values reserved.

**CurrentRMS Object** ()

Average flow of charge through a conductor.

**multiplier attribute** *(PowerOfTenMultiplierType)*

Specifies exponent of value.

**value attribute** *(Uint16)*

Value in amperes RMS (uom 5)

**FixedPointType Object** ()

Abstract type for specifying a fixed-point value without a given unit of measure.

**multiplier attribute** *(PowerOfTenMultiplierType)*

Specifies exponent of uom.

**value attribute** *(Int16)*

Dimensionless value

**UnsignedFixedPointType Object** ()

Abstract type for specifying an unsigned fixed-point value without a given unit of measure.

**multiplier attribute** *(PowerOfTenMultiplierType)*

Specifies exponent of uom.

**value attribute** *(Uint16)*

Dimensionless value

**ActivePower Object** ()

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.

**multiplier attribute** *(PowerOfTenMultiplierType)*

Specifies exponent for uom.
value attribute (Int16)
Value in watts (uom 38)

AmpereHour Object ()
Available electric charge

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of uom.

value attribute (UInt16)
Value in ampere-hours (uom 106)

ApparentPower Object ()
The apparent power $S$ (in VA) is the product of root mean square (RMS) voltage and RMS current.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of uom.

value attribute (UInt16)
Value in volt-amperes (uom 61)

ReactivePower Object ()
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.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of uom.

value attribute (Int16)
Value in volt-amperes reactive (var) (uom 63)

FixedPowerFactor Object ()
Specifies a setpoint for Displacement Power Factor, the ratio between apparent and active powers at the fundamental frequency (e.g. 60 Hz).

displacement attribute (Int16)
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.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of 'displacement'.

FixedVAr Object ()
Specifies a signed setpoint for reactive power.

refType attribute (DERUnitRefType)
Indicates whether to interpret 'value' as %setMaxVAr or %statVArAvail.

value attribute (SignedPerCent)
Specify a signed setpoint for reactive power in % (see 'refType' for context).

WattHour Object ()
Active (real) energy
multiplier attribute (PowerOfTenMultiplierType)

Specifies exponent of uom.

value attribute (UInt16)

Value in watt-hours (uom 72)

VoltageRMS Object ()

Average electric potential difference between two points.

multiplier attribute (PowerOfTenMultiplierType)

Specifies exponent of uom.

value attribute (UInt16)

Value in volts RMS (uom 29)

ConnectStatusType Object ()

DER ConnectStatus value:

0 - N/A
1 - disconnected_unavail
2 - disconnected_avail
3 - connected_unavail
4 - connected_avail
5 - connected_on
6 - test_mode

All other values reserved.

dateTime attribute (TimeType)

The date and time at which the state applied.

value attribute (UInt8)

The value indicating the state.

InverterStatusType Object ()

DER InverterStatus value:

0 - N/A
1 - off
2 - sleeping (auto-shutdown) or DER is at low output power/voltage
3 - starting up or ON but not producing power
4 - tracking MPPT power point
5 - forced power reduction/derating
6 - shutting down
7 - one or more faults exist
8 - standby (service on unit) - DER may be at high output voltage/power
9 - test mode
10 - as defined in manufacturer status
All other values reserved.

dateTime attribute (TimeType)
The date and time at which the state applied.

value attribute (UInt8)
The value indicating the state.

LocalControlModeStatusType Object ()
DER LocalControlModeStatus/value:
0 – local control
1 – remote control
All other values reserved.

dateTime attribute (TimeType)
The date and time at which the state applied.

value attribute (UInt8)
The value indicating the state.

ManufacturerStatusType Object ()
DER ManufacturerStatus/value: String data type

dateTime attribute (TimeType)
The date and time at which the state applied.

value attribute (String6)
The value indicating the state.

OperationalModeStatusType Object ()
DER OperationalModeStatus value:
0 - Not applicable / Unknown
1 - Off
2 - Operational mode
3 - Test mode
All other values reserved.

dateTime attribute (TimeType)
The date and time at which the state applied.

value attribute (UInt8)
The value indicating the state.

StateOfChargeStatusType Object ()
DER StateOfChargeStatus value: Percent data type

dateTime attribute (TimeType)
The date and time at which the state applied.
value attribute (PerCent)

The value indicating the state.

StorageModeStatusType Object ()

DER StorageModeStatus value:

0 – storage charging
1 – storage discharging
2 – storage holding

All other values reserved.

dateTime attribute (TimeType)

The date and time at which the state applied.

value attribute (UInt8)

The value indicating the state.

15.1.22 Links Package

Contains definitions of Link specializations used to require certain associations.

AccountBalanceLink Object (Link)

SHALL contain a Link to an instance of AccountBalance.

ActiveBillingPeriodListLink Object (ListLink)

SHALL contain a Link to a List of active BillingPeriod instances.

ActiveCreditRegisterListLink Object (ListLink)

SHALL contain a Link to a List of active CreditRegister instances.

ActiveDERControlListLink Object (ListLink)

SHALL contain a Link to a List of active DERControl instances.

ActiveEndDeviceControlListLink Object (ListLink)

SHALL contain a Link to a List of active EndDeviceControl instances.

ActiveFlowReservationListLink Object (ListLink)

SHALL contain a Link to a List of active FlowReservation instances.

ActiveProjectionReadingListLink Object (ListLink)

SHALL contain a Link to a List of active ProjectionReading instances.

ActiveSupply Interruption Override ListLink Object (ListLink)

SHALL contain a Link to a List of active Supply Interruption Override instances.

ActiveTargetReadingListLink Object (ListLink)

SHALL contain a Link to a List of active TargetReading instances.

ActiveTextMessageListLink Object (ListLink)

SHALL contain a Link to a List of active TextMessage instances.

ActiveTimeTariffIntervalListLink Object (ListLink)

SHALL contain a Link to a List of active TimeTariffInterval instances.

AssociatedDERProgramListLink Object (ListLink)

SHALL contain a Link to a List of DERPrograms having the DERControl(s) for this DER.
**AssociatedUsagePointLink Object** (Link)
SHALL contain a Link to an instance of UsagePoint. If present, this is the submeter that
monitors the DER output.

**BillingPeriodListLink Object** (ListLink)
SHALL contain a Link to a List of BillingPeriod instances.

**BillingReadingListLink Object** (ListLink)
SHALL contain a Link to a List of BillingReading instances.

**BillingReadingSetListLink Object** (ListLink)
SHALL contain a Link to a List of BillingReadingSet instances.

**ConfigurationLink Object** (Link)
SHALL contain a Link to an instance of Configuration.

**ConsumptionTariffIntervalListLink Object** (ListLink)
SHALL contain a Link to a List of ConsumptionTariffInterval instances.

**CreditRegisterListLink Object** (ListLink)
SHALL contain a Link to a List of CreditRegister instances.

**CustomerAccountLink Object** (Link)
SHALL contain a Link to an instance of CustomerAccount.

**CustomerAccountListLink Object** (ListLink)
SHALL contain a Link to a List of CustomerAccount instances.

**CustomerAgreementListLink Object** (ListLink)
SHALL contain a Link to a List of CustomerAgreement instances.

**DemandResponseProgramLink Object** (Link)
SHALL contain a Link to an instance of DemandResponseProgram.

**DemandResponseProgramListLink Object** (ListLink)
SHALL contain a Link to a List of DemandResponseProgram instances.

**DERAvailabilityLink Object** (Link)
SHALL contain a Link to an instance of DERAvailability.

**DERCapabilityLink Object** (Link)
SHALL contain a Link to an instance of DERCapability.

**DefaultDERControlLink Object** (Link)
SHALL contain a Link to an instance of DefaultDERControl. This is the default mode of the
DER which MAY be overridden by DERControl events.

**DERControlListLink Object** (ListLink)
SHALL contain a Link to a List of DERControl instances.

**DERCurveLink Object** (Link)
SHALL contain a Link to an instance of DERCurve.

**DERCurveListLink Object** (ListLink)
SHALL contain a Link to a List of DERCurve instances.
**DERLink Object** (Link)  
SHALL contain a Link to an instance of DER.

**DERListLink Object** (ListLink)  
SHALL contain a Link to a List of DER instances.

**DERProgramLink Object** (Link)  
SHALL contain a Link to an instance of DERProgram.

**DERProgramListLink Object** (ListLink)  
SHALL contain a Link to a List of DERProgram instances.

**DERSettingsLink Object** (Link)  
SHALL contain a Link to an instance of DERSettings.

**DERStatusLink Object** (Link)  
SHALL contain a Link to an instance of DERStatus.

**DeviceCapabilityLink Object** (Link)  
SHALL contain a Link to an instance of DeviceCapability.

**DeviceInformationLink Object** (Link)  
SHALL contain a Link to an instance of DeviceInformation.

**DeviceStatusLink Object** (Link)  
SHALL contain a Link to an instance of DeviceStatus.

**EndDeviceControlListLink Object** (ListLink)  
SHALL contain a Link to a List of EndDeviceControl instances.

**EndDeviceLink Object** (Link)  
SHALL contain a Link to an instance of EndDevice.

**EndDeviceListLink Object** (ListLink)  
SHALL contain a Link to a List of EndDevice instances.

**FileLink Object** (Link)  
This element MUST be set to the URI of the most recent File being loaded/activated by the LD.  
In the case of file status 0, this element MUST be omitted.

**FileListLink Object** (ListLink)  
SHALL contain a Link to a List of File instances.

**FileStatusLink Object** (Link)  
SHALL contain a Link to an instance of FileStatus.

**FlowReservationRequestListLink Object** (ListLink)  
SHALL contain a Link to a List of FlowReservationRequest instances.

**FlowReservationResponseListLink Object** (ListLink)  
SHALL contain a Link to a List of FlowReservationResponse instances.

**FunctionSetAssignmentsListLink Object** (ListLink)  
SHALL contain a Link to a List of FunctionSetAssignments instances.

**HistoricalReadingListLink Object** (ListLink)  
SHALL contain a Link to a List of HistoricalReading instances.
IPAddrListLink Object (ListLink)
SHALL contain a Link to a List of IPAddr instances.

IPInterfaceListLink Object (ListLink)
SHALL contain a Link to a List of IPInterface instances.

LLInterfaceListLink Object (ListLink)
SHALL contain a Link to a List of LLInterface instances.

LoadShedAvailabilityLink Object (Link)
SHALL contain a Link to an instance of LoadShedAvailability.

LogEventListLink Object (ListLink)
SHALL contain a Link to a List of LogEvent instances.

MessagingProgramListLink Object (ListLink)
SHALL contain a Link to a List of MessagingProgram instances.

MeterReadingLink Object (Link)
SHALL contain a Link to an instance of MeterReading.

MeterReadingListLink Object (ListLink)
SHALL contain a Link to a List of MeterReading instances.

MirrorUsagePointListLink Object (ListLink)
SHALL contain a Link to a List of MirrorUsagePoint instances.

NeighborListLink Object (ListLink)
SHALL contain a Link to a List of Neighbor instances.

NotificationListLink Object (ListLink)
SHALL contain a Link to a List of Notification instances.

PowerStatusLink Object (Link)
SHALL contain a Link to an instance of PowerStatus.

PrepaymentLink Object (Link)
SHALL contain a Link to an instance of Prepayment.

PrepaymentListLink Object (ListLink)
SHALL contain a Link to a List of Prepayment instances.

PrepayOperationStatusLink Object (Link)
SHALL contain a Link to an instance of PrepayOperationStatus.

PriceResponseCfgListLink Object (ListLink)
SHALL contain a Link to a List of PriceResponseCfg instances.

ProjectionReadingListLink Object (ListLink)
SHALL contain a Link to a List of ProjectionReading instances.

RateComponentLink Object (Link)
SHALL contain a Link to an instance of RateComponent.

RateComponentListLink Object (ListLink)
SHALL contain a Link to a List of RateComponent instances.
ReadingLink Object (Link)
A Link to a Reading.

ReadingListLink Object (ListLink)
SHALL contain a Link to a List of Reading instances.

ReadingSetListLink Object (ListLink)
SHALL contain a Link to a List of ReadingSet instances.

ReadingTypeLink Object (Link)
SHALL contain a Link to an instance of ReadingType.

RegistrationLink Object (Link)
SHALL contain a Link to an instance of Registration.

ResponseListLink Object (ListLink)
SHALL contain a Link to a List of Response instances.

ResponseSetListLink Object (ListLink)
SHALL contain a Link to a List of ResponseSet instances.

RPLInstanceListLink Object (ListLink)
SHALL contain a Link to a List of RPLInterface instances.

RPLSourceRoutesListLink Object (ListLink)
SHALL contain a Link to a List of RPLSourceRoutes instances.

SelfDeviceLink Object (Link)
SHALL contain a Link to an instance of SelfDevice.

ServiceSupplierLink Object (Link)
SHALL contain a Link to an instance of ServiceSupplier.

SubscriptionListLink Object (ListLink)
SHALL contain a Link to a List of Subscription instances.

SupplyInterruptionOverrideListLink Object (ListLink)
SHALL contain a Link to a List of SupplyInterruptionOverride instances.

SupportedLocaleListLink Object (ListLink)
SHALL contain a Link to a List of SupportedLocale instances.

TargetReadingListLink Object (ListLink)
SHALL contain a Link to a List of TargetReading instances.

TariffProfileLink Object (Link)
SHALL contain a Link to an instance of TariffProfile.

TariffProfileListLink Object (ListLink)
SHALL contain a Link to a List of TariffProfile instances.

TextMessageListLink Object (ListLink)
SHALL contain a Link to a List of TextMessage instances.

TimeLink Object (Link)
SHALL contain a Link to an instance of Time.
TimeTariffIntervalListLink Object  (ListLink)
SHALL contain a Link to a List of TimeTariffInterval instances.

UsagePointLink Object  (Link)
SHALL contain a Link to an instance of UsagePoint.

UsagePointListLink Object  (ListLink)
SHALL contain a Link to a List of UsagePoint instances.
### Appendix C – Examples and Guidelines [INFORMATIVE]

The contents of this section are not considered to be normative and are provided for reference. Text contained in { } in the following examples are considered to be placeholders for the actual text or value.

#### 16.1 Registration: Remote

This flow diagram describes client device remote registration to the customer HAN. Note that these are application layer details only, agnostic to the various layer 2 network joining techniques used by various PHY/MAC.

![Remote Registration Flow Diagram](image)

**Figure 16-1: Remote Registration**

Note: An XML example in POX encoding is provided along with the EXI equivalent.

**Table 16-1 POX Example: Registration Remote.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(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).</td>
</tr>
<tr>
<td>2</td>
<td>Client device joins the HAN (layer 2).</td>
</tr>
<tr>
<td>3</td>
<td>Client issues DNS-SD request to locate its EndDevice (keyed by its SFDI).</td>
</tr>
<tr>
<td>4</td>
<td>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.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>5</td>
<td>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.</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>Server verifies client identity because client certificate hashes to the client SFDI.</td>
</tr>
<tr>
<td>8</td>
<td>Client GETs its EndDevice from Server as returned in DNS-SD Discovery</td>
</tr>
<tr>
<td></td>
<td>Client sends the following request:</td>
</tr>
<tr>
<td></td>
<td>GET /edev/3 HTTP/1.1</td>
</tr>
<tr>
<td></td>
<td>Host: {hostname}</td>
</tr>
<tr>
<td></td>
<td>Accept: application/sep+xml</td>
</tr>
<tr>
<td>9</td>
<td>Server responds with the EndDevice resource.</td>
</tr>
<tr>
<td></td>
<td>Server sends the following response:</td>
</tr>
<tr>
<td></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td></td>
<td>Content-Type: application/sep+xml</td>
</tr>
<tr>
<td></td>
<td>Content-Length: {contentLength}</td>
</tr>
<tr>
<td></td>
<td>&lt;EndDevice href=&quot;/edev/3&quot; subscribable=&quot;0&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ConfigurationLink href=&quot;/edev/3/cfg&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;DeviceInformationLink href=&quot;/edev/3/di&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;DeviceStatusLink href=&quot;/edev/3/ds&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;DeviceStatusLink href=&quot;/edev/3/fs&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;PowerStatusLink href=&quot;/edev/3/ps&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;sFDI&gt;987654321005&lt;/sFDI&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;FunctionSetAssignmentsListLink all=&quot;3&quot; href=&quot;/edev/3/fsal&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;RegistrationLink href=&quot;/edev/3/reg&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;SubscriptionListLink all=&quot;0&quot; href=&quot;/edev/3/subl&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/EndDevice&gt;</td>
</tr>
<tr>
<td>10</td>
<td>Client GETs its Registration (containing its PIN) from the Server as found in the EndDevice Resource.</td>
</tr>
<tr>
<td></td>
<td>Client sends the following request:</td>
</tr>
<tr>
<td></td>
<td>GET /edev/3/reg HTTP/1.1</td>
</tr>
<tr>
<td></td>
<td>Host: {hostname}</td>
</tr>
<tr>
<td></td>
<td>Accept: application/sep+xml</td>
</tr>
<tr>
<td>11</td>
<td>Server responds with the Registration resource. Note: the PIN is thus transmitted over a secure channel.</td>
</tr>
<tr>
<td></td>
<td>Server sends the following response:</td>
</tr>
<tr>
<td></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td></td>
<td>Content-Type: application/sep+xml</td>
</tr>
<tr>
<td></td>
<td>Content-Length: {contentLength}</td>
</tr>
<tr>
<td></td>
<td>&lt;Registration href=&quot;/edev/3/reg&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;pIN&gt;123455&lt;/pIN&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Registration&gt;</td>
</tr>
<tr>
<td>12</td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 16-2 EXI Example: Registration Remote.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(same as POX example)</td>
</tr>
<tr>
<td>2</td>
<td>(same as POX example)</td>
</tr>
<tr>
<td>3</td>
<td>(same as POX example)</td>
</tr>
</tbody>
</table>
| 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.  
TXT Record: level=+S0  
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). |
| 5    | (same as POX example) |
| 6    | (same as POX example) |
| 7    | Client GETs its EndDevice from Server as returned in DNS-SD Discovery.  
Client sends the following request:  
GET /edev/3 HTTP/1.1  
Host: {hostname}  
Accept: application/sep-exi; level=-S0  
Note: level=-S0 indicates the client can receive EXI with SEP 2.0 schema without any arbitrary elements and attributes. |
| 8    | Server responds with the EndDevice resource.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep-exi  
Content-Length: {contentLength}  
a030 114c c26a 0049 7b2b 232b b179 9800  
0034 bd95 9195 d8bc ccbd 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 b332 b300 0038  
bd95 9195 d8bc ccbd cdd5 89b0  
(124-bytes in binary, shown in hexadecimal) |
| 9    | Client GETs its Registration (containing its PIN) from the Server as found in the EndDevice Resource.  
Client sends the following request:  
GET /edev/3/reg HTTP/1.1  
Host: {hostname}  
Accept: application/sep-exi; level=-S0 |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Server responds with the Registration resource. Note: the PIN is thus transmitted over a secure channel.</td>
</tr>
<tr>
<td>10.2</td>
<td>Server sends the following response:</td>
</tr>
<tr>
<td></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td></td>
<td>Content-Type: application/sep-exi</td>
</tr>
<tr>
<td></td>
<td>Content-Length: {contentLength}</td>
</tr>
<tr>
<td></td>
<td>a030 114c c2ef 034b d959 195d 8bcc cbdc 9959 cb96 00</td>
</tr>
<tr>
<td></td>
<td>(21-bytes in binary, shown in hexadecimal)</td>
</tr>
<tr>
<td>11.1</td>
<td>(same as POX example)</td>
</tr>
</tbody>
</table>

### 16.2 Registration: Local

This flow diagram describes client device local registration to the customer HAN. Note that these are application layer details only, agnostic to the various layer 2 network joining techniques used by various PHY/MAC.

![RegistrationLocalDiagram](image-url)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Client device joins the HAN (layer 2).</td>
</tr>
<tr>
<td>2</td>
<td>Client issues DNS-SD request to locate its EndDevice (keyed by its SFDI).</td>
</tr>
<tr>
<td>3</td>
<td>Client does not find a desired SFDI response.</td>
</tr>
<tr>
<td>4</td>
<td>Client issues DNS-SD request to locate any 'smartenergy' server.</td>
</tr>
<tr>
<td>5</td>
<td>A Server or multiple servers provides DNS-SD responses. If no reply is found then no 'smartenergy' servers are present on the network.</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>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</td>
</tr>
</tbody>
</table>
| 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} 

```xml
<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"/>
    <SubscriptionListLink all="0" href="/edev/3/subl"/>
  </EndDevice>
</EndDeviceList>
```
<p>| 9    | Client does not find its SFDI in any of the EndDevices in the EndDeviceList. |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10   | Client does a POST to EndDeviceList to add its EndDevice to the server.  
Client sends the following request:  
POST /edev HTTP/1.1  
Host: {hostname}  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  
<EndDevice href="/edev/3" xmlns="http://zigbee.org/sep">  
  <sFDI>987654321005</sFDI>  
</EndDevice> |
| 11   | Server indicates the EndDevice Record was created with the following:  
If new EndDevice entry is created, the Server would respond with the following where Location indicates the path to the newly created EndDevice resource:  
HTTP/1.1 201 Created  
Location: /edev/4 |
| 12   | Server verifies client identity because client certificate hashes to the client SFDI. |
### 16.3 Discovery: Function Set Assignment

**Figure 16-3: Discovery: Function Set Assignment**

Note: The Client finds a Server with a validated EndDevice record. The EndDevice Record is found to have a FunctionSetAssignmentListLink with a FunctionSetAssignment record that contains the desired Resource.

**Table 16-4 POX Example: Discovery Function Set Assignment**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>2</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>3</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>4</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>5</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 6    | Server responds with the EndDevice resource with a FunctionSetAssignmentListLink. Server sends the following response: HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength}  
```xml
<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>
``` |
| 7    | (See Registration Examples) |
| 8    | (See Registration Examples) |
| 9    | (See Registration Examples) |
| 10   | Client GETs its FunctionSetAssignment from the Server as found in the EndDevice Resource. Client sends the following request: GET /edev/3/fsal?l=2 HTTP/1.1 Host: {hostname} Accept: application/sep+xml  
Server responds with the FunctionSetAssignment resource. Server sends the following response: HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength}  
```xml
<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/fsa1/0/drppl"/>
    <MessagingProgramListLink all="1" href="/edev/3/fsa1/0/msgl"/>
    <mRID>0ED30F5A0000</mRID>
    <description>FunctionSetAssignment Suave</description>
  </FunctionSetAssignments>
  <FunctionSetAssignments href="/edev/3/fsal/1" subscribable="0">
    <MessagingProgramListLink all="1" href="/edev/3/fsa1/1/msgl"/>
    <mRID>0ED30F5A0001</mRID>
    <description>FunctionSetAssignment Guttural</description>
  </FunctionSetAssignments>
</FunctionSetAssignmentsList>
``` |
| 11   | Client GETs its desired Resource from the Server as found in the FunctionSetAssignment Resource. In this case, the Resource of choice is a DemandResponseProgram. (see DemandResponse Examples) |
| 12   | (see DemandResponse Examples) |
16.4 **Discovery: Without Function Set Assignment**

**Figure 16-4: Discovery Without Function Set Assignment**

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.

**Table 16-5 POX Example: Discovery Without Function Set Assignment.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>2</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>3</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>4</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>5</td>
<td>(See Registration Examples)</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 6 | 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.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>
```

7 | (See Registration Examples) |
8 | (See Registration Examples) |
9 | (See Registration Examples) |
10 | Client GETs its DeviceCapabilities from the Server as found in the DNS-SD TXT Record.  
Client sends the following request:  
GET /dcap HTTP/1.1  
Host: [hostname]  
Accept: application/sep+xml  

11 | Server responds with the DeviceCapabilities resource.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>
```

12 | Client GETs its desired Resource from the Server as found in the DeviceCapabilities Resource. In this case, the Resource of choice is a DemandResponseProgram.  
(see DemandResponse Examples) |
13 | (see DemandResponse Examples) |
16.5 Discovery: Undirected Without Function Set Assignment

Note: The Client discovers all Servers and searches through the records to find a Server with a validated EndDevice record for itself. The EndDevice Record is not found to have a FunctionSetAssignmentListLink with a FunctionSetAssignment record that contains the desired Resource. The Client searches for the desired Resource in EndDevice or in DeviceCapabilities.

Table 16-6 POX Example: Discovery Undirected Without Function Set Assignment.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(See Registration: Local Example)</td>
</tr>
<tr>
<td>2</td>
<td>(See Registration: Local Example)</td>
</tr>
<tr>
<td>3</td>
<td>(See Registration: Local Example)</td>
</tr>
<tr>
<td>4</td>
<td>(See Registration: Local Example)</td>
</tr>
</tbody>
</table>
### 16.6 Subscription / Notification

![Subscription/Notification Diagram]

Note: An XML example in POX encoding is provided along with the EXI equivalent.
### Table 16-7 POX Example: Subscription / Notification.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | After discovering the URI of the server's SubscriptionList and desired Resource, the Client can append new subscription entries to the list.  
**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.  
**POST /edev/8/sub HTTP/1.1**  
Host: {hostname}  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>
```
| 2    | If new subscription entries are created, the Server would respond with the following, where Location indicates the path to the newly created Subscription resource:  
**HTTP/1.1 201 Created**  
Location: /edev/8/sub/5  
If entries were not created but modified the Server would respond with:  
**HTTP/1.1 204 No Content**  
| 3    | A change on the subscribed resource occurs which satisfies the above specified Conditions.  
| 4    | Notification is sent from the Server to the Client:  
**POST /note HTTP/1.1**  
Host: {hostname}  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>
```
| 5    | The Client responds with HTTP Response:  
**HTTP/1.1 201 Created**  

### Table 16-8 EXI Example: Subscription / Notification.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | (same as POX example with the following exceptions)  
POST format is set with  
*Content-Type: application/sep-exi*  
Notification encoding type is specified with:  
<encoding>1</encoding>  
Notification schema version and options are specified with  
<level>-S0</level>  
POST /edev/8/sub HTTP/1.1  
Host: {hostname}  
Content-Type: application/sep-exi  
Content-Length: {contentLength}  
```plaintext
a030 114c c30c 41e5 eeae 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  
*Content-Type: application/sep-exi*  
POST /note HTTP/1.1  
Host: {hostname}  
Content-Type: application/sep-exi  
Content-Length: {contentLength}  
```plaintext
a030 114c c2b5 41e5 eeae 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) |
16.7 Demand Response: General

Table 16-9 POX Example: Demand Response – General.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 2    | Client GETs the list of DemandResponsePrograms from the DRLC Server. Client sends the following request:  
GET /drp?l=2 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3    | DemandResponse server responds with the list of DemandResponsePrograms.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml  
<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="1" href="/drp/2/edc"/>  
    <primacy>1</primacy>  
  </DemandResponseProgram>  
</DemandResponseProgramList>  
```
| 4    | Client GETs the list of EndDeviceControls from the DRLC Server for the desired DemandResponseProgram.  
Client sends the following request:  
GET /drp/1/edc HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>DemandResponse server responds with the list of EndDeviceControls. &lt;br&gt;Server sends the following response: &lt;br&gt;HTTP/1.1 200 OK &lt;br&gt;Content-Type: application/sep+xml &lt;br&gt;Content-Length: {contentLength} &lt;br&gt;&lt;EndDeviceControlList all=&quot;1&quot; results=&quot;1&quot; subscribable=&quot;0&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt; &lt;EndDeviceControl href=&quot;/drp/1/edc&quot; replyTo=&quot;{hostname}/rsp&quot; responseRequired=&quot;01&quot; subscribable=&quot;0&quot;&gt; &lt;mRID&gt;CAFEFEED&lt;/mRID&gt; &lt;description&gt;Emergency One Hour Coffee Brew&lt;/description&gt; &lt;creationTime&gt;1234556&lt;/creationTime&gt; &lt;EventStatus&gt; &lt;currentStatus&gt;0&lt;/currentStatus&gt; &lt;dateTime&gt;1234556&lt;/dateTime&gt; &lt;potentiallySuperseded&gt;false&lt;/potentiallySuperseded&gt; &lt;reason&gt;Need Caffeine Soon&lt;/reason&gt; &lt;/EventStatus&gt; &lt;interval&gt; &lt;duration&gt;360&lt;/duration&gt; &lt;start&gt;1234900&lt;/start&gt; &lt;/interval&gt; &lt;randomizeDuration&gt;60&lt;/randomizeDuration&gt; &lt;randomizeStart&gt;60&lt;/randomizeStart&gt; &lt;deviceCategory&gt;08&lt;/deviceCategory&gt; &lt;drProgramMandatory&gt;true&lt;/drProgramMandatory&gt; &lt;loadShiftForward&gt;true&lt;/loadShiftForward&gt; &lt;SetPoint&gt; &lt;heatingSetpoint&gt;10000&lt;/heatingSetpoint&gt; &lt;/SetPoint&gt; &lt;/EndDeviceControl&gt; &lt;/EndDeviceControlList&gt;</td>
</tr>
<tr>
<td>6</td>
<td>For each EndDeviceControl with ResponseRequired Bit 0, set the Client POSTs a Response with a Status of &quot;Event Received&quot; to the Response resource specified by the replyTo field in the EndDeviceControl. &lt;br&gt;Client sends the following: &lt;br&gt;POST /rsp HTTP/1.1 &lt;br&gt;Host: {hostname} &lt;br&gt;Content-Type: application/sep+xml &lt;br&gt;Content-Length: {contentLength} &lt;br&gt;&lt;Response xmlns=&quot;http://zigbee.org/sep&quot;&gt; &lt;createdDateTime&gt;1234560&lt;/createdDateTime&gt; &lt;endDeviceLFDI&gt;C0FFEE00&lt;/endDeviceLFDI&gt; &lt;status&gt;1&lt;/status&gt; &lt;subject&gt;CAFEFEED&lt;/subject&gt; &lt;/Response&gt;</td>
</tr>
<tr>
<td>7</td>
<td>Response Server replies with Status 201 Created: &lt;br&gt;HTTP/1.1 201 Created</td>
</tr>
<tr>
<td>8</td>
<td>Client begins the event at the defined start Time.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 9    | Client POST a Response with a Status of "Event Started" to the Response resource specified by the replyTo field in the EndDeviceControl.  
Client sends the following:  
POST /rsp HTTP/1.1  
Host: {hostname}  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml  
<Response xmlns="http://zigbee.org/sep">  
    <createdDateTime>1234900</createdDateTime>  
    <endDeviceLFDI>C0FFEE00</endDeviceLFDI>  
    <status>2</status>  
    <subject>CAFEFEED</subject>  
</Response>  
```
| 10   | Response Server Responds with Status 201 Created:  
HTTP/1.1 201 Created |
| 11   | Client completes the event. |
| 12   | Client POST a Response with a Status of "Event Completed" to the Response resource specified by the replyTo field in the EndDeviceControl.  
Client sends the following:  
POST /rsp HTTP/1.1  
Host: {hostname}  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml  
<Response xmlns="http://zigbee.org/sep">  
    <createdDateTime>1235260</createdDateTime>  
    <endDeviceLFDI>C0FFEE00</endDeviceLFDI>  
    <status>3</status>  
    <subject>CAFEFEED</subject>  
</Response>  
```
| 13   | Response Server Responds with Status 201 Created:  
HTTP/1.1 201 Created |
16.8 **Demand Response: Cancel**

**Figure 16-8: Demand Response - Cancel**

**Table 16-10 POX Example: Demand Response - Cancel**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>2</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>3</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>4</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>5</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>6</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>7</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>8</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>9</td>
<td>(Same as Demand Response: General)</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>10</td>
<td>(Same as Demand Response: General)</td>
</tr>
</tbody>
</table>
| 11   | 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. Client sends the following request:  
GET /drp/1/edc/3 HTTP/1.1  
Host: [hostname]  
Accept: application/sep+xml |
| 12   | DemandResponse Server indicates the event is cancelled:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>
```
| 13   | Client cancels the event. |
| 14   | Client responds with the specific instance of the EndDeviceControl with an updated EventStatus of 'Canceled'  
Server sends the following response:  
POST /rsp HTTP/1.1  
Host: [hostname]  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<Response xmlns="http://zigbee.org/sep">  
<createdDateTime>1235100</createdDateTime>  
<endDeviceLFDI>C0FFEE00</endDeviceLFDI>  
<status>6</status>  
<subject>CAFEFEED</subject>  
</Response>
```
| 15   | Response Server replies with Status 201 Created:  
HTTP/1.1 201 Created |
16.9 Distributed Energy Resource: General

Figure 16-9: Distributed Energy Resource: General
Table 16-11 POX Example: Distributed Energy Resource - General.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 2    | 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:  

```
GET /derp HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml; level=+S0
```

| 3    | The DER server responds with the requested DERProgram list, for example:  

```
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>
```

| 4    | The client locates or creates a local Notification resource that can be used to receive notification of changes to the DERControlList. |
| 5    | 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.  

```
POST /edev/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>
```

| 6    | The DER server responds with Status 201 Created. In the future, the DER server will "push" DERControlList updates to the client.  

```
HTTP/1.1 201 CREATED  
Location: /edev/0/sub/1  
Content-Length: 0
```

| 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:  

```
GET /derp/0/derc?s=0&l=1 HTTP/1.1  
Host: {hostname}
```
8 DER server responds with the first DERControl on the list. Server sends the following response:

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>
      <currentTime>1</currentTime>
      <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>

9 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:

GET /derp/0/dc/3 HTTP/1.1
Host: {hostname}
Accept: application/sep+xml; level=+S0

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>
    <xvalue>99</xvalue>
    <yvalue>50</yvalue>
  </CurveData>
  <CurveData>
  
10 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.

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>
  

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:

```xml
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>
```

Response Server responds with:
HTTP/1.1 201 Created

Client begins the event at the specified start (or current) time.

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:

```xml
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>
```

Response Server responds with:
HTTP/1.1 201 Created

Client completes the event.

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:

```xml
POST /rsp HTTP/1.1
Host: {hostname}
Content-Type: application/sep+xml
Content-Length: {contentLength}
```
Response Server responds with:

HTTP/1.1 201 Created

16.10 Metering: Reading

This is a use case where a metering function set client (e.g., In-Premises Display) queries a reading from a usage point. For this example, we will assume the meter is configured for 4 TOU tiers and no Blocks and that we want to read the current tier 3 consumption.
Figure 16-10: Meter Reading

Note: In most cases, registration is required to obtain access to metering.

Table 16-12 POX Example: Meter Reading.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the UsagePointList from the Metering Server  
   Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.  
   Client sends the following request:  
   GET /upt HTTP/1.1  
   Host: {hostname}  
   Accept: application/sep+xml |
## Step 2

Metering Server replies with **UsagePointList**.

A typical response looks like:

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>
```

## Step 3

Client GETs the **MeterReadingList** from the Metering Server.

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.

Client sends the following request:

```
GET /upt/0/mr?s=0&l=10 HTTP/1.1
Host: {hostname}
Accept: application/sep+xml
```

## Step 4

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.

A typical response looks like:

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>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5    | Client GETs the ReadingType from the Metering Server.  
      | 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. |
| 6    | Metering Server replies with ReadingType.  
      | A typical response looks like:  
      | HTTP/1.1 200 OK  
      | Content-Type: application/sep+xml  
      | Content-Length: {contentLength} |
| 7    | Client GETs the ReadingSetList from the Metering Server.  
      | 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. |
| 8    | Metering Server replies with ReadingSetList with the ReadingSet of interest.  
      | A typical response looks like:  
<pre><code>  | HTTP/1.1 200 OK |
</code></pre>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 9    | Client GETs the ReadingList from the Metering Server.  
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.  
Client sends the following request:  
GET /upt/0/mr/0/rs/0/r?s=3&l=12 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
| 10   | Metering Server replies with ReadingList with the Reading of interest.  
A typical response looks like:  
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>  
</Reading>  
</ReadingList> |
### 16.11 Metering: Interval

This is a use case where a metering function set client (e.g., In-Premises Display) queries for a specific set of interval readings from a usage point. For this example, we will assume that the meter is configured for 5 minute intervals.
Note: In most cases, registration is required to obtain access to metering.

Table 16-13 POX Example: Metering Interval.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the UsagePointList from the Metering Server.  
      | Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.  
      | Client sends the following request:  
      | GET /upt HTTP/1.1  
      | Host: (hostname)  
<pre><code>  | Accept: application/sep+xml |
</code></pre>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2    | Metering Server replies with UsagePointList. A typical response looks like:  
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">  
/MeterPoint>  
</UsagePointList>  |
| 3    | Client GETs the MeterReadingList from the Metering Server.  
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.  
Client sends the following request:  
GET /upt/0/mr?s=0&l=10 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml  |
| 4    | 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.  
A typical response looks like:  
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"/>  

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5    | Client GETs the ReadingType from the Metering Server.  
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.  
Client sends the following request:  
GET /upt/0/mr/0/rt?r=s=0&l=1 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
| 6    | Metering Server replies with ReadingType.  
A typical response looks like:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<ReadingType 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>
```

Note: Once the desired ReadingType is identified we proceed to the next step. |
| 7    | Client GETs the ReadingSetList from the Metering Server.  
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.  
Client sends the following request:  
GET /upt/0/mr/2/rs?s=0&l=4 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 8    | Metering Server replies with ReadingSetList with the ReadingSet of interest.  
     | A typical response looks like:  
     | 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>  
     | </ReadingSetList>  
| 9    | Once the ReadingSet that encompasses the starting time is identified, the client would GET the ReadingList from the Metering Server.  
     | Note: To identify the interval that has the desired start time the client would GET the reading set.  
     | Client sends the following request:  
     | GET /upt/0/mr/2/rs/4/r?s=0&l=12 HTTP/1.1  
     | Host: {hostname}  
     | Accept: application/sep+xml  
| 10   | Metering Server replies with ReadingList.  
     | A typical response looks like:  
     | HTTP/1.1 200 OK  
     | Content-Type: application/sep+xml  
<pre><code> | Content-Length: {contentLength} |
</code></pre>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>&lt;ReadingList all=&quot;12&quot; href=&quot;/upt/0/mr/2/rs/4/r&quot; results=&quot;12&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/0&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;00&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/2&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1162&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;01&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/4&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;02&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/6&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;03&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/8&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;04&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/a&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;05&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/c&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1162&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;06&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/e&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;07&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/10&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;08&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/12&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;09&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/14&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1162&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;0A&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;Reading href=&quot;/upt/0/mr/2/rs/4/r/16&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;value&gt;1163&lt;/value&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;localID&gt;0B&lt;/localID&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/Reading&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/ReadingList&gt;</code></td>
</tr>
</tbody>
</table>

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.
16.12 **Metering: Instantaneous**

This is a use case where a metering function set client (e.g., In-Premises Display) queries an instantaneous Watts reading from a usage point.

**Figure 16-12: Metering Instantaneous**

Note: In most cases, registration is required to obtain access to metering.

**Table 16-14 POX Example: Metering Instantaneous.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the UsagePointList from the Metering Server  
      Note: If directed through FunctionsSetAssignments to a particular UsagePoint, these first two steps would be skipped.  
      Client sends the following request:  
      ```
      GET /upt HTTP/1.1  
      Host: {hostname}  
      Accept: application/sep+xml  
      ``` |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2    | **Metering Server replies with UsagePointList.**  
A typical response looks like:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>```
| 3    | **Client GETs the MeterReadingList from the Metering Server.**  
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.  
Client sends the following request:  
GET /upt/0/mr?s=0&l=10 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml  

| 4    | **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.**  
A typical response looks like:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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"/>  
</MeterReadingList>```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5    | **Client GETs the ReadingType from the Metering Server.**  
  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.  
  Client sends the following request:  
  ```  
  GET /upt/0/mr/4/rt HTTP/1.1  
  Host: {hostname}  
  Accept: application/sep+xml  
  ``` |
| 6    | **Metering Server replies with ReadingType.**  
  A typical response looks like:  
  ```html  
  HTTP/1.1 200 OK  
  Content-Type: application/sep+xml  
  Content-Length: {contentLength}  
  ```  
  ```xml  
  <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>  
  ```  
  Note: Once the desired ReadingType is identified we proceed to the next step. |
| 7    | **Client GETs the Reading from the Metering Server.**  
  Note: Because the instantaneous value is in the resource indicated in the ReadingLink of MeterReading we can read that resource.  
  Client sends the following request:  
  ```  
  GET /upt/0/mr/4/r HTTP/1.1  
  Host: {hostname}  
  Accept: application/sep+xml  
  ```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Metering Server replies with the Reading. A typical response looks like: HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: {contentLength} &lt;Reading href=&quot;/upt/0/mr/4/r&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt; &lt;value&gt;14&lt;/value&gt; &lt;/Reading&gt; 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.</td>
</tr>
</tbody>
</table>

16.13 **Metering: Mirroring**

This is a use case where a mirror metering function set client (e.g., a gas meter) POSTs first its general information and then its data. It also shows a client of the gas meter data retrieving the most recent 24 intervals of data. An assumption is made that prior to this sequence the mirror client has discovered the URI of the appropriate Meter Mirroring Server.
Figure 16-13: Meter Mirroring

Note: In most cases, registration is required to obtain access to metering and meter mirroring.

Table 16-15 POX Example: Meter Mirroring.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 1    | 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.  
  
  Note: It passes two ReadingType definitions, one for Summation and one for Interval data.  
  
  Client sends the following request:  
  
  ```
  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>
  ``` |
| 2    | 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.  
  
  A typical response looks like:  
  
  ```
  HTTP/1.1 201 Created  
  Content-Length: 0  
  Location: /mup/0
  ``` |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Meter Mirroring Client POSTs a MirrorMeterReading to the Mirror Metering Server.</td>
</tr>
</tbody>
</table>

Client sends the following request:

```
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>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4    | Mirror Metering Server creates a MirrorMeterReading with the data supplied in the MirrorUsagePoint and then replies with the URI of the MirrorMeterReading. A typical response looks like:  
HTTP/1.1 201 Created  
Content-Length: 0  
Location: /upt/1/mr  
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    | Mirror Data Client GETs the UsagePointList from the Metering Server |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6    | Mirror Metering Server replies with UsagePointList. A typical response looks like:  
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> |
| 7    | Mirror Data Client GETs the MeterReadingList from the Mirror Metering Server. Note: We will choose /upt/1 because its role flags indicate it is a mirror.  
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.  
Client sends the following request:  
GET /upt/1/mr?s=0&l=10 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
| 8    | 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.  
A typical response looks like:  
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>  
  </MeterReading>  
  <ReadingLink href="/upt/1/mr/0/r"/>  
</MeterReadingList> |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| **9** | **Mirror Data** Client GETs the ReadingType from the Mirror Metering Server.  
**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.  
Client sends the following request:  
GET /upt/1/mr/1/rt HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml  
Mirror Metering Server replies with ReadingType.  
A typical response looks like:  
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>  
**Note:** Once the desired ReadingType is identified we proceed to the next step. |
| **10** | **Mirror Data** Client GETs the ReadingSetList from the Metering Server.  
**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.  
Client sends the following request:  
GET /upt/1/mr/1/rs?s=0&l=1 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml  
Mirror Metering Server replies with ReadingSetList with the ReadingSet of interest.  
A typical response looks like:  
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>  
</ReadingSet>  
</ReadingSetList>  
**Mirror Data** GETs the ReadingSetList from the Metering Server.  
**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. |
| **11** | **Mirror Data** Client GETs the ReadingSetList from the Metering Server.  
**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.  
Client sends the following request:  
GET /upt/1/mr/1/rs?s=0&l=1 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml  
Mirror Metering Server replies with ReadingSetList with the ReadingSet of interest.  
A typical response looks like:  
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>  
</ReadingSet>  
</ReadingSetList> }

Page 296  
Copyright © 2013, ZigBee Alliance, Inc. and HomePlug Powerline Alliance, Inc. All rights reserved.
Mirror Data Client GETs the ReadingList from the Mirror Metering Server.

Client sends the following request:

GET /upt/1/mr/1/rs/32/r?s=0&l=24 HTTP/1.1
Host: {hostname}
Accept: application/sep+xml

Mirror Metering Server replies with ReadingList with the Reading of interest.

A typical response looks like:

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"
xmns="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>
  </Reading>
</ReadingList>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;localID&gt;0A&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/f&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;5&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;0B&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/10&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;8&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;0C&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/11&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;9&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;0D&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/12&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;0E&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/13&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;12&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;0F&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/14&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;14&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;10&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/15&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;13&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;11&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/16&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;11&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;12&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/17&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;7&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;13&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/0&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;8&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;14&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/1&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;15&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/2&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;16&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading href=&quot;/upt/1/mr/1/rs/32/r/3&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/localID&gt;17&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ReadingList&gt;</td>
</tr>
</tbody>
</table>

15 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.

Client sends the following request:

POST /mup/0 HTTP/1.1
Host: {hostname}
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
|      | 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>  
|      | <MirrorMeterReading>  
|      |     <mRID>0900006CC8</mRID>  
|      |     <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>  
|      | </MirrorMeterReadingList>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;8&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;0C&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;9&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;0D&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;0E&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;12&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;0F&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;14&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;10&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;13&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;11&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;11&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;12&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;7&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;13&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;8&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;14&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;15&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;16&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;value&gt;10&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;localID&gt;17&lt;/localID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Reading&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/MirrorReadingSet&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/MirrorMeterReading&gt;</td>
</tr>
</tbody>
</table>

| 16   | Mirror Metering Server copies the data supplied into the corresponding UsagePoint and then replies with the URI of the MeterReadingList of that UsagePoint. |

A typical response looks like:

HTTP/1.1 201 Created
Content-Length: 0
Location: /upt/1/mr
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 17   | **Mirror Data**
      | **Client GETs the ReadingSetList from the Mirror Metering Server.**
      | **Note:** Because the Client cached the URI of the reading set list, it can skip ahead to this step.
      | Client sends the following request:
      | `GET /upt/1/mr/1/rs?s=0&l=1 HTTP/1.1`  
      | Host: {hostname}  
      | Accept: application/sep+xml |
| 18   | **Mirror Metering Server replies with ReadingSetList with the ReadingSet of interest.**
      | A typical response looks like:
      | 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>
      |   </ReadingSet>
      | </ReadingSetList>
      | ``` |
| 19   | **Mirror Data Client GETs the ReadingList from the Mirror Metering Server.**
      | Client sends the following request:
      | `GET /upt/1/mr/1/rs/33/r?s=0&l=24 HTTP/1.1`  
      | Host: {hostname}  
      | Accept: application/sep+xml |
| 20   | **Mirror Metering Server replies with ReadingList with the Reading of interest.**
      | A typical response looks like:
      | 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>
      | </ReadingList>
      | ```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
<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>
</Reading>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
|      | <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>  

7827 16.14 **Pricing: Time of Use**

7828 This flow diagram describes pricing client device obtaining pricing information.
Note: In most cases, registration is required to obtain access to pricing, and the client is directed to a specific TariffProfile through a Link in the FunctionSetAssignments found in their EndDevice FunctionSetAssignmentsListLink, or from a different function set such as Billing.

Table 16-16 POX Example: Pricing TOU.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the TariffProfile from the Pricing Server. Client sends the following request:  
GET /tp/3 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2    | Pricing server responds with the TariffProfile.  
Server sends the following response:  
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/3">  
  <serviceCategoryKind>0</serviceCategoryKind>  
</TariffProfile> |
| 3    | Client GETs the RateComponentList from the Pricing Server.  
Client sends the following request:  
GET /tp/3/rc?l=1 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
| 4    | Pricing server responds with the RateComponentList.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  
<RateComponentList all="1" href="/tp/3/rc/3" 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>  
    </ActiveTimeTariffIntervalListLink>  
    <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> |
| 5    | Client GETs the ReadingType from the Pricing Server.  
Client sends the following request:  
GET /rt/1 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
| 6    | Pricing server responds with the ReadingType.  
Server sends the following response: |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Client GETs the TimeTariffIntervalList from the Pricing Server.</td>
</tr>
<tr>
<td></td>
<td>Client sends the following request:</td>
</tr>
<tr>
<td></td>
<td>GET /tp/3/rc/3/tti?l=5 HTTP/1.1</td>
</tr>
<tr>
<td></td>
<td>Host: {hostname}</td>
</tr>
<tr>
<td></td>
<td>Accept: application/sep+xml</td>
</tr>
<tr>
<td>8</td>
<td>Pricing server responds with the TimeTariffIntervalList.</td>
</tr>
<tr>
<td></td>
<td>Server sends the following response:</td>
</tr>
<tr>
<td></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td></td>
<td>Content-Type: application/sep+xml</td>
</tr>
<tr>
<td></td>
<td>Content-Length: {contentLength}</td>
</tr>
</tbody>
</table>

```
<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>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;duration&gt;14400&lt;/duration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;start&gt;1357545600&lt;/start&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeDuration&gt;300&lt;/randomizeDuration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeStart&gt;300&lt;/randomizeStart&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ConsumptionTariffIntervalListLink all=&quot;1&quot; href=&quot;/tp/3/rc/3/tti/6/cti&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;touTier&gt;2&lt;/touTier&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/TimeTariffInterval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;TimeTariffInterval href=&quot;/tp/3/rc/3/tti/7&quot; subscribable=&quot;1&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;mRID&gt;63eed7b30c1c87a40000e566&lt;/mRID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;description&gt;On-Peak&lt;/description&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;creationTime&gt;1357430400&lt;/creationTime&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;EventStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;currentStatus&gt;0&lt;/currentStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;dateTime&gt;1357430400&lt;/dateTime&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;potentiallySuperseded&gt;false&lt;/potentiallySuperseded&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/EventStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;duration&gt;21600&lt;/duration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;start&gt;1357552800&lt;/start&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeDuration&gt;300&lt;/randomizeDuration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeStart&gt;300&lt;/randomizeStart&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ConsumptionTariffIntervalListLink all=&quot;1&quot; href=&quot;/tp/3/rc/3/tti/7/cti&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;touTier&gt;3&lt;/touTier&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/TimeTariffInterval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;TimeTariffInterval href=&quot;/tp/3/rc/3/tti/8&quot; subscribable=&quot;1&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;mRID&gt;9b04f0713e9212d90000e566&lt;/mRID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;description&gt;Mid-Peak 2&lt;/description&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;creationTime&gt;1357430400&lt;/creationTime&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;EventStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;currentStatus&gt;0&lt;/currentStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;dateTime&gt;1357430400&lt;/dateTime&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;potentiallySuperseded&gt;false&lt;/potentiallySuperseded&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/EventStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;duration&gt;18000&lt;/duration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;start&gt;1357574400&lt;/start&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeDuration&gt;300&lt;/randomizeDuration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeStart&gt;300&lt;/randomizeStart&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ConsumptionTariffIntervalListLink all=&quot;1&quot; href=&quot;/tp/3/rc/3/tti/8/cti&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;touTier&gt;2&lt;/touTier&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/TimeTariffInterval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;TimeTariffInterval href=&quot;/tp/3/rc/3/tti/9&quot; subscribable=&quot;1&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;mRID&gt;c13c8755dc39b5950000e566&lt;/mRID&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;description&gt;Off-Peak 2&lt;/description&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;creationTime&gt;1357430400&lt;/creationTime&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;EventStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;currentStatus&gt;0&lt;/currentStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;dateTime&gt;1357430400&lt;/dateTime&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;potentiallySuperseded&gt;false&lt;/potentiallySuperseded&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/EventStatus&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;duration&gt;10800&lt;/duration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;start&gt;1357592400&lt;/start&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/interval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeDuration&gt;300&lt;/randomizeDuration&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;randomizeStart&gt;300&lt;/randomizeStart&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ConsumptionTariffIntervalListLink all=&quot;1&quot; href=&quot;/tp/3/rc/3/tti/9/cti&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;touTier&gt;1&lt;/touTier&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/TimeTariffInterval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/TimeTariffIntervalList&gt;</td>
</tr>
</tbody>
</table>

Client GETs a ConsumptionTariffIntervalList from the Pricing Server (note that there is a
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ConsumptionTariffIntervalList for each TimeTariffInterval, but only one is shown below.)</td>
</tr>
<tr>
<td></td>
<td>Client sends the following request:</td>
</tr>
<tr>
<td></td>
<td>GET /tp/3/rc/3/tti/5/cti?l=1 HTTP/1.1</td>
</tr>
<tr>
<td></td>
<td>Host: {hostname}</td>
</tr>
<tr>
<td></td>
<td>Accept: application/sep+xml</td>
</tr>
<tr>
<td>10</td>
<td>Pricing server responds with the ConsumptionTariffIntervalList.</td>
</tr>
<tr>
<td></td>
<td>Server sends the following response:</td>
</tr>
<tr>
<td></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td></td>
<td>Content-Type: application/sep+xml</td>
</tr>
<tr>
<td></td>
<td>Content-Length: {contentLength}</td>
</tr>
<tr>
<td></td>
<td>&lt;ConsumptionTariffIntervalList all=&quot;1&quot; href=&quot;/tp/3/rc/3/tti/5/cti&quot; results=&quot;1&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;ConsumptionTariffInterval href=&quot;/tp/3/rc/3/tti/5/cti/1&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;consumptionBlock&gt;1&lt;/consumptionBlock&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;price&gt;113000&lt;/price&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;startValue&gt;0&lt;/startValue&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ConsumptionTariffInterval&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/ConsumptionTariffIntervalList&gt;</td>
</tr>
</tbody>
</table>
16.15 **Billing: Billing Period**

This flow diagram describes a billing client device obtaining billing information including billing period.

![Billing: Billing Period Flow Diagram](image)

**Figure 16-15: Billing Period**

Note: In most cases, registration is required to obtain access to billing, and the client is directed to a specific `CustomerAccount` through a Link in the `FunctionSetAssignments` found in their `EndDevice FunctionSetAssignmentsListLink`.

**Table 16-17 POX Example: Billing Period.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the CustomerAccount from the Billing Server.  
Client sends the following request:  
GET /ca/1 HTTP/1.1  
Host: [hostname]  
Accept: application/sep+xml |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2    | Billing server responds with the CustomerAccount.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml  
<CustomerAccount 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>  
```
| 3    | Client GETs the ServiceSupplier from the Billing Server.  
Client sends the following request:  
GET /ss HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml
| 4    | Billing server responds with the ServiceSupplier.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml  
<ServiceSupplier 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>  
```
| 5    | Client GETs the CustomerAgreementList from the Billing Server.  
Client sends the following request:  
GET /bill/1/ca HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6    | Billing server responds with the CustomerAgreementList. <p>Server sends the following response:</p> <p>HTTP/1.1 200 OK</p> <p>Content-Type: application/sep+xml</p> <p>Content-Length: {contentLength}</p> <p>&lt;CustomerAgreementList all="1" href="/bill/1/ca" results="1" subscribable="0" xmlns="http://zigbee.org/sep">&lt;CustomerAgreement href="/bill/1/ca/1">
   &lt;mRID>65f839fc951345e70000e566</mRID>
   &lt;description&gt;Electric Service - 4/1/2012&lt;/description&gt;
   &lt;ActiveBillingPeriodListLink all="1" href="/bill/1/ca/1/actbp"/>
   &lt;BillingPeriodListLink all="1" href="/bill/1/ca/1/bp"/>
   &lt;HistoricalReadingListLink all="1" href="/bill/1/ca/1/ver"/>
   &lt;ProjectionReadingListLink all="1" href="/bill/1/ca/1/pro"/>
   &lt;serviceLocation&gt;Acct. XXX-XXXXX-384 (Elm St.)&lt;/serviceLocation&gt;
   &lt;TariffProfileLink href="/tp/3"/>
   &lt;UsagePointLink href="/upt/1"/>
   &lt;/CustomerAgreement&gt;
&lt;/CustomerAgreementList&gt;</p> |
| 7    | Client GETs the BillingPeriodList from the Billing Server. <p>Client sends the following request:</p> <p>GET /bill/1/ca/1/bp?l=1 HTTP/1.1</p> <p>Host: {hostname}</p> <p>Accept: application/sep+xml</p> |
| 8    | Billing server responds with the BillingPeriodList. <p>Server sends the following response:</p> <p>HTTP/1.1 200 OK</p> <p>Content-Type: application/sep+xml</p> <p>Content-Length: {contentLength}</p> <p>&lt;BillingPeriodList all="2" href="/bill/1/ca/1/bp" results="1" subscribable="1" xmlns="http://zigbee.org/sep">&lt;BillingPeriod href="/bill/1/ca/1/bp">
   &lt;billLastPeriod&gt;140730000&lt;/billLastPeriod&gt;
   &lt;billToDate&gt;83550000&lt;/billToDate&gt;
   &lt;interval&gt;
      &lt;duration&gt;2419200&lt;/duration&gt;
      &lt;start&gt;1360195200&lt;/start&gt;
   &lt;/interval&gt;
   &lt;statusTimeStamp&gt;1361577600&lt;/statusTimeStamp&gt;
   &lt;/BillingPeriod&gt;</p> |
16.16 **Billing: Historical**

This flow diagram describes a billing client device obtaining historical billing readings.

![Flow diagram](Image)

**Table 16-18 POX Example: Billing Historical.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the HistoricalReadingList from the Billing Server.  
Client sends the following request:  
GET /bill/1/ca/1/ver HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml | |
| 2    | Billing server responds with the HistoricalReadingList.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  
<HistoricalReadingList all="1" href="/bill/1/ca/1/ver" results="1"
### Step 3
Client GETs the ReadingType from the Billing Server.

Client sends the following request:

```
GET /rt/3 HTTP/1.1
Host: {hostname}
Accept: application/sep+xml
```

### Step 4
Billing server responds with the ReadingType.

Server sends the following response:

```
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>
```

### Step 5
Client GETs the BillingReadingSetList from the Billing Server.

Client sends the following request:

```
GET /bill/1/ca/1/ver/1/brs?l=1 HTTP/1.1
Host: {hostname}
Accept: application/sep+xml
```

### Step 6
Billing server responds with the BillingReadingSetList.

Server sends the following response:

```
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">
  <mRID>82866ecc81c9638a0000e566</mRID>
  <timePeriod>
    <duration>86400</duration>
    <start>1361491200</start>
  </timePeriod>
  <BillingReadingSet href="/bill/1/ca/1/ver/1/brs/1" />
</BillingReadingSetList>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7    | Client GETs the BillingReadingList from the Billing Server.  
Client sends the following request:  
GET /bill/1/ca/1/ver/1/brs/1/br?l=1 HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |
| 8    | Billing server responds with the BillingReadingList.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>  
    <Charge>  
      <kind>0</kind>  
      <value>429400</value>  
    </Charge>  
  </BillingReading>
</BillingReadingList>
```
16.17 **Billing: Projection**

This flow diagram describes a billing client device obtaining a billing projection.

![Billing: Projection Diagram]

**Figure 16-17: Billing Projection**

**Table 16-19 POX Example: Billing Projection.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Client GETs the ProjectionReadingList from the Billing Server. Client sends the following request:  
GET /bill/1/ca/1/pro HTTP/1.1  
Host: {hostname}  
Accept: application/sep+xml |  
| 2    | Billing server responds with the ProjectionReadingList. Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: {contentLength}  

```xml
<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>d0b05a2e65144fca0000e566</mRID>  
<description>Billing Projections</description>  
</ProjectionReading>  
</ProjectionReadingList>
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3    | Client GETs the BillingReadingSetList from the Billing Server.  
Client sends the following request:  
GET /bill/1/ca/1/pro/1/brs?l=1 HTTP/1.1  
Host: (hostname)  
Accept: application/sep+xml |
| 4    | Billing server responds with the BillingReadingSetList.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: [contentLength]  

```xml
<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>
```
| 5    | Client GETs the BillingReadingList from the Billing Server.  
Client sends the following request:  
GET /bill/1/ca/1/pro/1/brs/1/br?l=1 HTTP/1.1  
Host: (hostname)  
Accept: application/sep+xml |
| 6    | Billing server responds with the BillingReadingList.  
Server sends the following response:  
HTTP/1.1 200 OK  
Content-Type: application/sep+xml  
Content-Length: [contentLength]  

```xml
<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>
```
16.18 File Loading

The flow diagram below describes how an LD queries an FS for a list of available files, loads a file from the FS, and verifies and activates the loaded file. It also describes how the LD may optionally maintain status of the file loading operation and reporting of same to the FS. The flow assumes network joining and device registration with service provider have already occurred.

Figure 16-18: File Load - FlowDiagram

Step descriptions are listed below.
Table 16-20 File Load: Flow Description

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The LD issues discovery requests to locate available FS (DNS-SD subtype &quot;file&quot;).</td>
</tr>
<tr>
<td>2</td>
<td>The LD has discovered the FS and obtained the URL of its FileList.</td>
</tr>
</tbody>
</table>
| 3    | The LD queries the FileList to determine if there are any available files to be downloaded to the LD. An example FileList query:  
  
  GET http://host1/fileList?s=0&l=5&type=0x0000&mfId=37244&mfModel=123abc&mfVer=23.47.102  
  HTTP/1.1  
  Host: host1  
  
  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. |
| 4    | The FS responds to the LD with the FileList satisfying the query parameters. An example FS response:  
  
  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>  
  
  The FS reports that is 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.  
  
  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). |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>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). The LD will periodically issue the following request: GET <a href="http://host1/myFile1">http://host1/myFile1</a> HTTP/1.1 Host: host1</td>
</tr>
<tr>
<td>10</td>
<td>The FS responds with /myFile1. HTTP/1.1 200 OK Content-Type: application/sep+xml Content-Length: ...&lt;File href=&quot;http://host1/myFile1&quot; xmlns=&quot;http://zigbee.org/sep&quot;&gt; &lt;activateTime&gt;3763784682&lt;/activateTime&gt; &lt;fileURI&gt;<a href="http://host1/myfile1.bin">http://host1/myfile1.bin</a>&lt;/fileURI&gt; &lt;mfID&gt;37244&lt;/mfID&gt; &lt;mfModel&gt;123abc&lt;/mfModel&gt; &lt;mfVer&gt;23.48.1&lt;/mfVer&gt; &lt;size&gt;128000&lt;/size&gt; &lt;type&gt;00&lt;/type&gt;&lt;/File&gt; 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.</td>
</tr>
<tr>
<td>11</td>
<td>The LD waits until the activation time specified for /myFile1 is reached, then places the file into the activate state. In the case of a firmware file, the file is now the running image.</td>
</tr>
<tr>
<td>12</td>
<td>The LD again updates its FileStatus resource.</td>
</tr>
<tr>
<td>13</td>
<td>The LD PUTs its FileStatus resource to a remote EndDevice server. In this example, the EndDevice is hosted at the FS. Thus the FS is provided with a final status of the LD load operation.</td>
</tr>
</tbody>
</table>
16.19 **Flow Reservation: General**

The following is a summary of the example:

1. **Step 1** - Client (PEV) creates a FlowReservationRequest at 5:00 PM for charging between midnight and 8:00 AM, 12 kWh energy requested at a power level of 7 kW, 7371 seconds duration requested.

2. Subsequently, the Server creates a FlowReservationResponse with a charge interval between 1:00 AM and 5:20 AM at 3 kW.

3. **Step 3** - Client requests the FlowReservationResponseList to find the response matching the request.

4. **Step 4** – Server responds with the FlowReservationResponseList.

5. **Step 5** - Client periodically requests the FlowReservationResponse to look for changes.

6. **Step 7** - Client updates PowerStatus periodically during charging.

**Note:** In most cases, registration is required to obtain access to request flow reservations.
### Table 16-21 POX Example: Flow Reservation - General.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1 | **Client POSTs a FlowReservationRequest to the Flow Reservation Server at 9/22/2013 5:00 PM.**  
Client sends the following request:  

```xml
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>  
  <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>  
    <dateTime>1379869200</dateTime>  
    <requestStatus>0</requestStatus>  
  </RequestStatus>  
</FlowReservationRequest>
```

2 | **Flow Reservation server responds with the FlowReservationRequest location.**  
Server sends the following response:  

HTTP/1.1 201 Created  
Location: /edev/3/frq/1

3 | **Client GETs the FlowReservationResponseList from the Flow Reservation Server to look for a response to the request.**  
Client sends the following request:  

```http
GET /edev/3/frp HTTP/1.1  
Host: {hostname}
```

4 | **Flow Reservation server responds with the FlowReservationResponseList**  
Server sends the following response:  

HTTP/1.1 200 OK  
Content-Type: application/sep+xml
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5 | Client GETs the FlowReservationResponseList periodically (or subscribes) from the Flow Reservation Server.  
Client sends the following request:  
GET /edev/3/frp HTTP/1.1  
Host: (hostname) |
| 6 | Flow Reservation server responds with the FlowReservationResponseList.  
Server sends the following response:  
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>  
    <energyAvailable>  
      <multiplier>0</multiplier>  
      <value>12000</value>  
    </energyAvailable>  
    <powerAvailable>  
      <multiplier>0</multiplier>  
      <value>3000</value>  
    </powerAvailable>  
    <subject>68512866203db3b10000e566</subject>  
  </FlowReservationResponse>  
</FlowReservationResponseList> |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7    | From 1:00 AM to 5:20 AM while charging the client periodically updates its PowerStatus. Client sends the following request:  

```
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>  
```

| 8    | Flow Reservation server responds:  

```
HTTP/1.1 204 No Content  
```

| 7877 |
| 7878 |
16.20 **Flow Reservation: Cancel**

The following is a summary of the example:

**Step 1** - Client (PEV) creates a FlowReservationRequest at 5:00 PM for charging between midnight and 8:00 AM.
- Subsequently, the Server creates a FlowReservationResponse with a charge interval between 1:00 AM and 5:20 AM at 3 kW.

**Step 3** – At 1:00 AM the Client wants to change the time the charging is needed and therefore creates a new FlowReservationRequest for charging between 1:00 AM and 5:00 AM.
- Subsequently, the Server creates a FlowReservationResponse with a charge interval between 1:02 AM and 4:22 AM at 4 kW.

**Step 5** - Client cancels the original FlowReservationRequest.
- Subsequently, the Server cancels the original FlowReservationResponse.

**Step 7** - Client requests the FlowReservationResponseList.

**Step 8** – Server responds with the FlowReservationResponseList, the first FlowReservationResponse is canceled and the second FlowReservationResponse is Active.
Table 16-22 POX Example: Flow Reservation - Cancel

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Client POSTs a FlowReservationRequest to the Flow Reservation Server at 9/22/2013 5:00 PM. Client sends the following request:</td>
</tr>
</tbody>
</table>

POST /edev/3/frq HTTP/1.1
Host: (hostname)

<FlowReservationRequest xmlns="http://zigbee.org/sep">
  <mRID>68512866203db3b1000e566</mRID>
  <description>Charge between 12AM and 8AM</description>
  <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>
    <dateTime>1379869200</dateTime>
    <requestStatus>0</requestStatus>
  </RequestStatus>
</FlowReservationRequest>

2 Flow Reservation server responds with the FlowReservationRequest location.

Server sends the following response:

HTTP/1.1 201 Created
Location: /edev/3/frq/1

3 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.

Client sends the following request:

POST /edev/3/frq HTTP/1.1
Host: (hostname)

<FlowReservationRequest xmlns="http://zigbee.org/sep">
  <mRID>68512866203db3b1000e577</mRID>
  <description>Charge between 1AM and 5AM</description>
  <creationTime>1379898000</creationTime>
  <durationRequested>7371</durationRequested>
  <energyRequested>
    <multiplier>3</multiplier>
    <value>12</value>
  </energyRequested>
  <intervalRequested>
    <duration>28800</duration>
    <start>1379923600</start>
  </intervalRequested>
  <powerRequested>
    <multiplier>3</multiplier>
    <value>7</value>
  </powerRequested>
  <RequestStatus>
    <dateTime>1379923600</dateTime>
    <requestStatus>0</requestStatus>
  </RequestStatus>
</FlowReservationRequest>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
|      | <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>1379898060</dateTime>  
       | <!-- Requested -->  
       | <requestStatus>1</requestStatus>  
       | </RequestStatus>  
       | </FlowReservationRequest> |
| 4 | Flow Reservation server responds with the FlowReservationRequest location.  
Server sends the following response:  
HTTP/1.1 201 Created  
Location: /edev/3/frq/2 |
| 5 | Client PUTs a canceled status to the first FlowReservationRequest.  
Client sends the following request:  
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> |
| 6 | Flow Reservation server sends the following response:  
HTTP/1.1 204 No Content |
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7    | Client GETs the FlowReservationResponseList periodically (or subscribes) from the Flow Reservation Server.  

Client sends the following request:  

GET /edev/3/frp?l=2 HTTP/1.1  
Host: {hostname} |
| 8    | Flow Reservation server responds with the FlowReservationResponseList. The first FlowReservationResponse is Canceled, the second is Active.  

Server sends the following response:  

HTTP/1.1 200 OK  
Content-Type: application/sep+xml  

```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>  
    <creationTime>1379869260</creationTime>  
    <EventStatus>  
      <!-- Canceled -->  
      <currentStatus>2</currentStatus>  
      <dateTime>1379898060</dateTime>  
      <potentiallySuperseded>true</potentiallySuperseded>  
    </EventStatus>  
    <interval>  
      <!-- 4 hours 20 minutes -->  
      <duration>15600</duration>  
      <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>  
    <creationTime>1379898120</creationTime>  
    <EventStatus>  
      <!-- Active -->  
      <currentStatus>1</currentStatus>  
      <dateTime>1379898120</dateTime>  
      <potentiallySuperseded>false</potentiallySuperseded>  
    </EventStatus>  
    <interval>  
      <!-- 3 hours 20 minutes -->  
      <duration>12000</duration>  
      <start>1379898120</start>  
    </interval>  
  </FlowReservationResponse>  
</FlowReservationResponseList>  
```
16.21 Event Randomization

The following are examples of how to set the randomization parameters to accomplish different randomization strategies. The attributes of an event that define its behavior in time are, the "interval" that defines the "start" and "duration" of the event, "randomizeStart" and "randomizeDuration". The later two controlling the randomization behavior. For these examples, we will assume we are looking at DRLC events and that the events are causing reductions in load over a large population. The graphs are indicating the aggregated load for the entire population for a single event.

16.21.1 Simple Event

![Simple Event Diagram](image)

**Figure 16-21: Simple Event - No Ramp Up - No Ramp Down**

Note: This use case is NOT targeted for large population control.

**Table 16-23 Ramp Value.**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3600</td>
</tr>
<tr>
<td>randomizeStart</td>
<td>0</td>
</tr>
<tr>
<td>randomizeDuration</td>
<td>0</td>
</tr>
</tbody>
</table>
16.21.2 **Event with Positive Randomized Duration**

![Diagram](image)

**Figure 16-22: Event with Positive Randomization Duration**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3600</td>
</tr>
<tr>
<td>randomizeStart</td>
<td>0</td>
</tr>
<tr>
<td>randomizeDuration</td>
<td>300</td>
</tr>
</tbody>
</table>

16.21.3 **Event with Positive Randomized Start**

![Diagram](image)
Figure 16-23: Event with Positive Randomization Start

Table 16-25 Event Start Time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3600</td>
</tr>
<tr>
<td>randomizeStart</td>
<td>300</td>
</tr>
<tr>
<td>randomizeDuration</td>
<td>0</td>
</tr>
</tbody>
</table>

16.21.4 Event with Negative Randomized Start

Figure 16-24: Event with Negative Randomized Start

Table 16-26 Event Start Time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3600</td>
</tr>
<tr>
<td>randomizeStart</td>
<td>-300</td>
</tr>
<tr>
<td>randomizeDuration</td>
<td>0</td>
</tr>
</tbody>
</table>
16.21.5 Event with Positive Randomization and Finishing in one Hour

Ramp Down Positive, Ramp Up Negative

Figure 16-25: Positive Randomization in an Hour

Table 16-27 Event Start Time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3300</td>
</tr>
<tr>
<td>randomizeStart</td>
<td>300</td>
</tr>
<tr>
<td>randomizeDuration</td>
<td>0</td>
</tr>
</tbody>
</table>

16.21.6 Event with Negative Randomized Start and at Least One Hour Duration

Ramp Down Negative, Ramp Up Positive

Figure 16-26: Event with Negative Start in One Hour

Table 16-28 Event Start Time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3900</td>
</tr>
</tbody>
</table>
16.21.7 Event with Randomized Start and Long Ramp Up

Table 16-29 Event Start Time.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>3600</td>
</tr>
<tr>
<td>randomizeStart</td>
<td>300</td>
</tr>
<tr>
<td>randomizeDuration</td>
<td>300</td>
</tr>
</tbody>
</table>
16.21.8 **Event with Positive Randomized Start and Fixed End Time**

Ramp Down Positive (or Negative), No Ramp Up

**Figure 16-28: Randomized Start and Fixed End Time**

Note: This scenario is NOT possible with the current set of parameters. For grid stability "snap" back on is not a desirable use case.
17 Appendix D – Guidelines [INFORMATIVE]

17.1 Pricing Implementation Guidelines

This section discusses guidelines for implementing common service provider tariff rate designs and enabling HAN devices to match a TariffProfile.RateComponent.ReadingType with a meaningful Metering value in cases where the Pricing and Metering servers may not be coordinated or match. This section assumes readers are familiar with the Pricing and Metering function set text in early sections. Readers may also wish to consult the WADL and schema [ZB 13-0201] for more detailed information.

17.1.1 Implementing Common Tariff Designs

This section is intended to provide guidance to service providers and HAN server implementers on how to use the Pricing function set and its design flexibility in a way that encourages standard uses and practices. High level description of the resources and attribute values are provided here; for detailed XML examples of the pricing function set, see Section 16.14. The following four tariff rate designs are discussed further in the text below:

- Flat rate
- Time-of-use tiers
- Consumption-based blocks
- Time-of-use tiers with consumption-based blocks

All four designs will assume that pricing information is only provided for the forward direction (commodity delivered), and is applicable for any flow rate (demand). This corresponds to a TariffProfile with a single RateComponent in its RateComponentList. This RateComponent would have the flowRateStartLimit and flowRateEndLimit attributes elided, and would link to a ReadingType with the flowDirection attribute set to "1".

17.1.2 Flat Rate Design

The flat rate design assumes a constant price for a commodity over one or many billing periods. With no price differentiation based on the time of day or on the amount of the commodity already consumed, the flat-rate design requires only a single touTier and a single consumptionBlock. Therefore the tariff's RateComponent must link to a ReadingType with the following attributes:

- numberOfConsumptionBlocks = 1
- numberOfTouTiers = 1

Per [ZB 13-0201], the RateComponent links to a TimeTariffIntervalList. The flat rate design only requires a single TimeTariffInterval in this list, though information for future seasons could be communicated by providing additional TimeTariffIntervals in the list (only one TimeTariffInterval can be active at a given time). Any TimeTariffInterval instance provided under a flat rate design must have the touTier attribute set to "1".

Again per [ZB 13-0201], TimeTariffInterval links to a ConsumptionTariffIntervalList. For the flat rate design there must only be one ConsumptionTariffInterval in the list. This ConsumptionTariffInterval defines the consumption price per commodity unit. The startValue attribute in this ConsumptionTariffInterval must be "0" and the consumptionBlock attribute must be "1".
17.1.3 **Time-of-Use Rate Design**

Most TOU rate designs consist of a repeating series of time-differentiated rates. These can vary over the course of a day, and the series may be different on weekdays and weekends. Pricing servers may be implemented and configured with functionality that creates repeating TimeTariffIntervals based on an internal schedule in order to minimize backhaul network traffic, but this is out of scope for SEP 2. For the HAN, such a Pricing server would supply a steady stream of pricing "events" with a defined commencement and duration. Pricing service providers and their servers are encouraged to provide enough pricing information to cover the next 24-hour period since most consumer devices operate over this time scale. Pricing service providers are also encouraged to update their Pricing servers with the next sequential TimeTariffInterval instance at least four hours before its Effective Start Time.

For this TOU rate example, assume a daily three-tier (plus one Critical Peak Pricing (CPP) tier) rate design with the following characteristics as shown in Table 17-1:

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Description</th>
<th>Price</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midnight – 8 AM</td>
<td>Off-peak</td>
<td>$0.10</td>
<td>1</td>
</tr>
<tr>
<td>8 AM – 10 AM</td>
<td>Mid-peak</td>
<td>$0.20</td>
<td>2</td>
</tr>
<tr>
<td>10 AM – 6 PM</td>
<td>On-peak</td>
<td>$0.40</td>
<td>3</td>
</tr>
<tr>
<td>6 PM – Midnight</td>
<td>Mid-peak</td>
<td>$0.20</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CPP</td>
<td>$0.70</td>
<td>4</td>
</tr>
</tbody>
</table>

Tier numbers are ordered by price from lowest to highest (though note that the price attribute itself will be found in the ConsumptionTariffInterval resource linked to by a TimeTariffInterval, not in the TimeTariffInterval itself); this is mandatory in Smart Energy 2.0.

The TOU design provides price differentiation based on the time of day. However, as with the flat rate design, there is no price differentiation based on the amount of the commodity already consumed. In SEP 2 terms, this means that the TOU design requires two or more touTiers, but must provide only a single consumptionBlock. Using the above TOU example, the tariff's RateComponent must link to a ReadingType with the following attributes:

- numberOfConsumptionBlocks = 1
- numberOfTouTiers = 4

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.

In this scenario, the server's TimeTariffIntervalList will contain 8 TimeTariffIntervals: one expired event, one active event, and six scheduled events. Each TimeTariffInterval will link to a ConsumptionTariffIntervalList, where each list, under a TOU-only design, contains one ConsumptionTariffInterval. As with the flat rate design, each ConsumptionTariffInterval must have a consumptionBlock attribute of "1" and a startValue attribute of "0".
17.1.4 **Consumption-based Block Rate Design**

In some jurisdictions (e.g., California, British Columbia, the UK), consumption-based block rate designs have been put in place to incentivize efficient consumption of a commodity with prices that ratchet up over the course of the billing period based on the amount of the commodity consumed. In other scenarios, typically for commercial and industrial (C&I) customers, the price charged per commodity unit consumed falls after some usage threshold.

For this consumption-based rate design, assume a five-tier design with the following characteristics in Table 17-2:

<table>
<thead>
<tr>
<th>Block</th>
<th>Consumption startValue</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$0.12</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>$0.14</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>$0.23</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
<td>$0.27</td>
</tr>
<tr>
<td>5</td>
<td>350</td>
<td>$0.30</td>
</tr>
</tbody>
</table>

Block numbers are ordered by *startValue* from lowest to highest; this is mandatory in SEP 2. While the associated prices in this example also increase with each block number, this is not mandated by SEP 2.

The above example corresponds to a ReadingType with the following attributes:

- numberOfConsumptionBlocks = 5
- numberOfTouTiers = 1

As with the flat rate design, a server using the block rate design may only have one TimeTariffInterval in the TimeTariffIntervalList linked to from RateComponent. That TimeTariffInterval could correspond to an entire billing period or longer. However, unlike the flat rate design, that one TimeTariffInterval will link to a ConsumptionTariffIntervalList containing multiple (5, in this example) ConsumptionTariffInterval resources.

Note that under a block rate design, as opposed to flat or TOU rate designs, a pricing client cannot determine the active price solely from the pricing server. The client will also have to query the corresponding metering server to determine which consumption block is active.

17.1.5 **Combined Time-of-Use and Consumption-based Block Rate Design**

Yet other jurisdictions (e.g., California), have combined time-of-use and consumption-based rate designs to incentivize efficient consumption as well as shifting that consumption to times of lower system demand.

For this TOU with consumption-based rate design, assume a tariff structure that combines the previous two examples as shown in Table 17-3:

<table>
<thead>
<tr>
<th>Time</th>
<th>Tier</th>
<th>Block</th>
<th>Consumption startValue</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midnight – 8 AM</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Off-peak</td>
<td>$0.22</td>
</tr>
<tr>
<td>Time</td>
<td>Tier</td>
<td>Block</td>
<td>Consumption startValue</td>
<td>Description</td>
<td>Price</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>-------</td>
<td>------------------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>8 AM – 10 AM</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Mid-peak</td>
<td>$0.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>150</td>
<td></td>
<td>$0.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>250</td>
<td></td>
<td>$0.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>300</td>
<td></td>
<td>$0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>350</td>
<td></td>
<td>$0.50</td>
</tr>
<tr>
<td>10 AM – 6 PM</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>On-peak</td>
<td>$0.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>150</td>
<td></td>
<td>$0.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>250</td>
<td></td>
<td>$0.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>300</td>
<td></td>
<td>$0.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>350</td>
<td></td>
<td>$0.80</td>
</tr>
<tr>
<td>6 PM - Midnight</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Mid-peak</td>
<td>$0.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>150</td>
<td></td>
<td>$0.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>250</td>
<td></td>
<td>$0.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>300</td>
<td></td>
<td>$0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>350</td>
<td></td>
<td>$0.50</td>
</tr>
<tr>
<td>Intermittent</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>CPP</td>
<td>$0.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>150</td>
<td></td>
<td>$0.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>250</td>
<td></td>
<td>$0.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>300</td>
<td></td>
<td>$0.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>350</td>
<td></td>
<td>$1.00</td>
</tr>
</tbody>
</table>

Using the above TOU+Block example, the tariff’s RateComponent must link to a ReadingType with the following attributes:

- numberOfConsumptionBlocks = 5
- numberOfTouTiers = 4

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 is a CPP event scheduled for 1 PM – 3 PM on July 16, 2012.

In this scenario, the server’s TimeTariffIntervalList will contain 10 TimeTariffIntervals:

1) One expired event (Tier 1, present day 00:00:00 – 08:00:00)
2) One active event (Tier 2, present day 08:00:00 – 10:00:00)

3) Eight scheduled events, corresponding to:
   1) Tier 3, present day 10:00:00 – 13:00:00
   2) Tier 4 (CPP), present day 13:00:00 – 15:00:00
   3) Tier 3, present day 15:00:00 – 18:00:00
   4) Tier 2, present day 18:00:00 – next day 00:00:00
   5) Tier 1, next day 00:00:00 – 08:00:00
   6) Tier 2, next day 08:00:00 – 10:00:00
   7) Tier 3, next day 10:00:00 – 18:00:00
   8) Tier 2, next day 18:00:00 – day after next 00:00:00

Each TimeTariffInterval links to its own ConsumptionTariffIntervalList, and each ConsumptionTariffIntervalList will contain 5 ConsumptionTariffInterval resources with consumptionBlock, startValue, and price attributes set per the above TOU+Block rate.

As with the block rate design, under a TOU+Block design a pricing client cannot determine the active price solely from the pricing server; the client must determine the active block from the associated metering server.

### 17.1.6 Coordinated and Uncoordinated Pricing and Metering Servers

As with all function set servers in SEP 2, Pricing and Metering servers may be hosted by different service providers, served by the same provider on separate hosts, or coordinated or uncoordinated depending on the rules of the jurisdiction. This is problematic for certain tariff designs, particularly for consumption-based tariffs, which depend on usage accumulated during a given billing period in order to be able to provide accurate Pricing information. For example, in Texas, the Retail Energy Provider (REP) owns the customer relationship and is responsible for serving Pricing information, but the Transmission Distribution Service Provider (TDSP) owns the meter and provides one standard meter configuration for all users, regardless of the REP, which may change depending on the customer’s preferences. Another use case is a customer who may not have a smart meter but installs his own and links his HAN to his electricity service provider’s Pricing server on its website.

This section recommends mitigation strategies and coping mechanisms to help Pricing and Metering clients provide users with actionable information to make informed decisions in the event that the ReadingType instance referenced by the Pricing server does not match any of the Metering server implementations available in the HAN. Devices that follow Pricing servers primarily for price responsiveness (e.g., smart appliances) do not need to follow these recommendations because the numberOfTouTiers and touTier attributes in the pricing server’s ReadingType and TimeTariffInterval resources provide the guidance needed to optimize operational parameters.

The following rules are recommended for clients of Pricing and Metering resources and which aim to present a user with cost information about their usage or determine a nominal price for energy. These rules presume a client has performed service discovery and identified a suitable Metering server for its application (e.g., the premises aggregation point, PEV sub-meter). The first three rules assume that the Pricing and Metering servers are coordinated. Rules four and five are meant to guide implementations in scenarios where the Pricing and Metering servers are uncoordinated.
1) Metering servers SHOULD, at a minimum, serve a ReadingType instance for commodity delivered summation. This provides a functionality baseline that all devices can expect to be available in a HAN.

2) If Rule #1 is not satisfied, and if the ReadingTypeLink URI in the Pricing server's RateComponent object and the Metering server's MeterReading object are the same, then they are a match.

3) If Rule #2 is not satisfied, then the client SHOULD refer to the MeterReading instances' mRID attributes. If the Pricing and Metering servers' MeterReading.mRID attributes are the same, then they are a match.

4) If Rule #3 is not satisfied, then the client should compare values in order to find the best fit.
   a) The following attributes SHALL be exact matches in both servers' ReadingType instances:
      - commodity
      - flowDirection
      - kind
      - uom
   b) The following attributes SHOULD match in both servers' ReadingType instances or be accounted for (e.g., accumulationBehaviour attribute in Pricing server may be "Delta Data" whereas the Metering server may only provide "Cumulative": these can be matched with the appropriate device logic) or have at least one be designated as "Not Specified":
      - accumulationBehaviour
      - phase

5) If Rule #4 is not satisfied, clients SHOULD NOT display cost information to the user. Pricing clients SHOULD only display Pricing information. Pricing clients may indicate a mismatch between the Pricing and Metering server information and provide guidance to the user to contact his installer or service provider.

17.2 PEV Implementation Guidelines (subject to work with SAE and ISO / IEC)

Plug-In Electric Vehicles (PEVs) should implement Registration, DRLC, Pricing, and Messaging in order to alert the user of upcoming program events and to avoid charging during peak times. They may also implement FlowReservation in order to provide the ability to avoid high aggregated demand across multiple PEVs. They may also implement DER functionality in order to participate in programs that send advanced controls for grid conditioning and generation. Electric Vehicle Service Equipment (EVSE) may also use these functions, and may also use or implement metering to publish or obtain readings from another device in order to manage charging functions.