Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

IEEE STANDARDS ASSOCIATION

IEEE Standard for Smart Energy Profile Application Protocol

IEEE Communications Society

Sponsored by the
Power Line Communications Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 2030.5™-2018
(Revision of
IEEE Std 2030.5-2013)

IEEE Std 2030.5™-2018
(Revision of
IEEE Std 2030.5-2013)

IEEE Standard for Smart Energy Profile Application Protocol

Sponsor

Power Line Communications Committee
of the
IEEE Communications Society

Approved 7 May 2018

IEEE-SA Standards Board

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Abstract: The application layer with TCP/IP providing functions in the transport and Internet layers to enable utility management of the end user energy environment, including demand response, load control, time of day pricing, management of distributed generation, electric vehicles, etc. is defined in this standard. Depending on the physical layer in use (e.g., IEEE 802.15.4™, IEEE 802.11™, IEEE 1901™, IEEE 1901.2™), a variety of lower layer protocols may be involved in providing a complete solution. Generally, lower layer protocols are not discussed in this standard except where there is direct interaction with the application protocol. The mechanisms for exchanging application messages, the exact messages exchanged including error messages, and the security features used to protect the application messages are defined in this standard. With respect to the Open Systems Interconnection (OSI) network model, this standard is built using the four layer Internet stack model. The defined application profile sources elements from many existing standards, including IEC 61968 and IEC 61850, and follows a RESTful architecture (Fielding [B3]) using IETF protocols such as HTTP.

Keywords: adoption, application, application protocol, demand response, distributed energy resources, energy usage information, IEEE 2030.5™, load control, metering, plugin electric vehicles, prepayment, pricing communication, RESTful, SEP 2, smart energy, smart energy profile, Smart Energy Profile 2

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2018 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 21 December 2018. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

2030 is a registered trademark of The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-1-5044-4907-6 STD23130
Print: ISBN 978-1-5044-4908-3 STDPD23130

IEEE prohibits discrimination, harassment, and bullying.
For more information, visit http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html.
No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/IPR/disclaimers.html.

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at http://ieeexplore.ieee.org/ or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Participants

At the time this IEEE standard was completed, the Smart Energy Profile 2.0 Working Group had the following membership:

Robert Heile, *Chair*
Robby Simpson, *Vice Chair*

Electric Power Research Institute, Inc.
General Electric
Itron, Inc.
KITU Systems
Landis+Gyr
Sensus
SunSpec Alliance
Wi-SUN Alliance

The following members of the entity balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Electric Power Research Institute, Inc. — Tim Godfrey
General Electric — Robby Simpson
Itron, Inc. — Gene Falendysz
KITU Systems — Gordon Lum
Panasonic Corporation of North America — Michael Stelts
Power Plus Communications AG — Scott Willy
Sensus — Michael Cowan
Silver Spring Networks, Inc. — Don Sturek
Wi-SUN Alliance — Robert Heile

When the IEEE-SA Standards Board approved this standard on 7 May 2018, it had the following membership:

Jean-Philippe Faure, *Chair*
Vacant Position, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Chuck Adams
Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Doug Edwards
J. Travis Griffith
Gary Hoffman
Michael Janezic
Thomas Koshy
Joseph L. Koepfinger*
Kevin Lu
Daleep Mohla
Damir Novosel
Ronald C. Petersen
Annette D. Reilly
Robby Robson
Dorothy Stanley
Adrian Stephens
Mehmet Ulema
Phil Wennblom
Howard Wolfman
Yu Yuan

*Member Emeritus

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Introduction

This introduction is not part of IEEE Std 2030.5-2018™, IEEE Standard for Smart Energy Profile Application Protocol.

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 standard is intended to fulfill those needs.

The first publication of this standard (IEEE Std 2030.5-2013) was driven by, and sought to address the requirements of, many activities across the globe. Of note were 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.

This revision of IEEE Std 2030.5 was made with particular attention to the activities underway in California as part of the Rule 21 revision and the associated Smart Inverter Working Group as well as the revision of IEEE Std 1547™. This revision also seeks to address any errors and ambiguities discovered in the testing and deployment of the first publication.

This standard 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Contents

1. Overview 11
1.1 Scope 11
1.2 Purpose 11
1.3 Document organization....11
1.4 Requirement language 12
1.5 Typography conventions used 12
1.6 Design principles 12

2. Normative references....12

3. Definitions, acronyms, and abbreviations 14
3.1 Definitions 14
3.2 Acronyms and abbreviations 16

4. Design pattern....18
4.1 Protocol flexibility....18
4.2 General rules/best practices 18
4.3 WADL 19
4.4 Schema 19
4.5 Uniform resource identifiers....20
4.6 List resources....20
4.7 Resource design rules 24

5. Application support 24
5.1 Overview 24
5.2 Use of TCP 24
5.3 URI encoding....25
5.4 HTTP headers....25
5.5 HTTP response codes 27
5.6 Application payload syntax 30
5.7 Content negotiation....31

6. Security....32
6.1 Introduction 32
6.2 Security attributes 32
6.3 Device credentials....39
6.4 Resource access authentication and authorization context 40
6.5 Resource access authentication....42
6.6 Resource access authorization 42
6.7 Cipher suites 42
6.8 Default security policy....43
6.9 Registration....44
6.10 Security LogEvents....46
6.11 Certificate management 46

7. Discovery....59
7.1 Introduction 59
7.2 Service instance 60
7.3 Service name....61
7.4 TXT record 61
7.5 Subtype queries....62

Copyright © 2018 IEEE. All rights reserved.

7.6 Discovery procedure 64

8. Support resources 64
8.1 Introduction 64
8.2 Resource section outlines 65
8.3 Device Capabilities function set 66
8.4 Self Device function set 67
8.5 End Device function set 67
8.6 Function Set Assignments function set 69
8.7 Subscription/Notification function set 70
8.8 Response function set 73

9. Common resources 77
9.1 Introduction 77
9.2 Time function set 77
9.3 Device Information function set 79
9.4 Power Status function set 79
9.5 Network Status function set 80
9.6 Log Event function set 81
9.7 Configuration function set 83
9.8 File Download function set 83

10. Smart energy resources 88
10.1 Introduction 88
10.2 Common application functionality 88
10.3 Demand Response and Load Control function set 95
10.4 Metering function set 99
10.5 Pricing function set 105
10.6 Messaging function set 109
10.7 Billing function set 111
10.8 Prepayment function set 114
10.9 Flow Reservation function set 116
10.10 Distributed Energy Resources function set 117
10.11 Metering Mirror function set 128

11. Manufacturer-specific proprietary extensions 130
11.1 Overview 130
11.2 xmDNS/DNS-SD 130
11.3 URIs 130
11.4 Resources 130
11.5 DeviceCapabilities resource 131

Annex A (informative) Web-application description language (WADL) 132
A.1 Introduction 132
A.2 Support resources section 132
A.3 Common resources section 136
A.4 Smart energy resources section 139

Annex B (informative) IEEE 2030.5 model 153
B.1 Introduction 153
B.2 IEEE 2030.5 package 153

Annex C (informative) Examples and guideline 267
C.1 Introduction 267
C.2 Registration: Remote 267
C.3 Registration: Local 270

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.4 Discovery: Function Set Assignment 272
C.5 Discovery: Without Function Set Assignment 274
C.6 Discovery: Undirected without Function Set Assignment 276
C.7 Subscription/Notification 277
C.8 Demand response: General 280
C.9 Demand response: Cancel 284
C.10 Distributed energy resource: General 286
C.11 Metering: Reading 290
C.12 Metering: Interval 295
C.13 Metering: Instantaneous 302
C.14 Metering: Mirroring 305
C.15 Pricing: Time of use 317
C.16 Billing: Billing period 322
C.17 Billing: Historical 324
C.18 Billing: Projection 328
C.19 File loading 330
C.20 Flow Reservation: General 334
C.21 Flow Reservation: Cancel 339
C.22 Event randomization 343

Annex D (informative) Guidelines 348
D.1 Pricing implementation guidelines 348
D.2 PEV implementation guidelines (subject to work with SAE and ISO/IEC) 353

Annex E (informative) Mapping to IEEE Std 1547-2018 354

Annex F (informative) Bibliography 359

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

IEEE Standard for Smart Energy Profile Application Protocol

1. Overview

1.1 Scope

This standard defines the application layer with TCP/IP providing functions in the transport and Internet layers to enable utility management of the end user energy environment, including demand response, load control, time of day pricing, management of distributed generation, electric vehicles, etc. Depending on the physical layer in use (e.g., IEEE 802.15.4™, IEEE 802.11™, IEEE 1901™, IEEE 1901.2™), a variety of lower layer protocols may be involved in providing a complete solution. Generally, lower layer protocols are not discussed in this standard except where there is direct interaction with the application protocol. This standard defines the mechanisms for exchanging application messages, the exact messages exchanged including error messages, and the security features used to protect the application messages. With respect to the Open Systems Interconnection (OSI) network model, this standard is built using the four layer Internet stack model. The defined application profile sources elements from many existing standards, including IEC 61968 and IEC 61850, and follows a RESTful architecture (Fielding [B3]) using IETF protocols such as HTTP.

1.2 Purpose

The purpose of this document is to define the application protocol to enable utility management of the end user energy environment, including demand response, load control, time of day pricing, management of distributed generation, electric vehicles, etc. The defined application profile sources elements from many existing standards, including IEC 61968 and IEC 61850, and follows a RESTful architecture (Fielding [B3]) using IETF protocols such as HTTP.

1.3 Document organization

The following documents comprise the definition of IEEE 2030.5 and all IEEE 2030.5 devices will be required to maintain compliance to these documents:

— IEEE Std 2030.5 (this document)

— IEEE 2030.5 XML Schema Definition (XSD) (sep.xsd in the supplemental material of IEEE Std 2030.5)

— IEEE 2030.5 WADL (sep_wadl.xml in the supplemental material of IEEE Std 2030.5)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

The IEEE 2030.5 XSD contains the definitions of the IEEE 2030.5 resources, attributes, and elements as well as their textual descriptions. The IEEE 2030.5 WADL contains the recommended URI structures and use of HTTP methods associated with these objects. In addition, an IEEE 2030.5 UML model (also contained in the supplemental material of IEEE Std 2030.5) has been utilized in the creation of the IEEE 2030.5 XSD. The IEEE 2030.5 UML model is an informative document. Informative textual extracts of the IEEE 2030.5 XSD and the IEEE 2030.5 WADL are contained in this document for convenience.

1.4 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 IETF RFC 2119.

1.5 Typography conventions used

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

1.6 Design principles

As IEEE Std 2030.5 follows a RESTful architecture (Fielding [B3]), several design principles are important for the standard, including:

- 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. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEC 61968, Application integration at electric utilities—System interfaces for distribution management—All parts.[1]

IEEE Std 2030.5™, Supplemental material (http://standards.ieee.org/downloads/2030.5/).[2]

IEEE Std 802.1AR™, IEEE Standard for Local and Metropolitan Area Networks—Secure Device Identity (http://www.ieee802.org/1/pages/802.1ar.html).

[1] IEC publications are available from the International Electrotechnical Commission (http://www.iec.ch) and the American National Standards Institute (http://www.ansi.org/).
[2] The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

Copyright © 2018 IEEE. All rights reserved.

IETF, Extended Multicast DNS (https://tools.ietf.org/html/draft-lynn-homenet-site-mdns-01).

IETF RFC 793, Transmission Control Protocol (http://tools.ietf.org/html/rfc793).

IETF RFC 1630, Universal Resource Identifiers in WWW (http://tools.ietf.org/html/rfc1630).

IETF RFC 2119, Key Words for Use in RFCs to Indicate Requirement Levels (http://tools.ietf.org/html/rfc2119).

IETF RFC 2818, HTTP Over TLS (http://tools.ietf.org/html/rfc2818).

IETF RFC 2863, The Interface MIB (https://tools.ietf.org/html/rfc2863).

IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax (https://tools.ietf.org/html/rfc3986).

IETF RFC 4108, Using Cryptographic Message Syntax (CMS) to Protect Firmware (http://tools.ietf.org/html/rfc4108).

IETF RFC 4193, Unique IPv6 Unicast Addresses (http://tools.ietf.org/html/rfc4193).

IETF RFC 4291, IP Version 6 Addressing Architecture (http://tools.ietf.org/html/rfc4291).

IETF RFC 5246, Transport Layer Security (TLS) Protocol Version 1.2 (http://tools.ietf.org/html/rfc5246).

IETF RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (http://tools.ietf.org/html/rfc5280).

IETF RFC 5480, Elliptic Curve Cryptography Subject Public Key Information (http://tools.ietf.org/html/rfc5480).

IETF RFC 6550, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. (http://tools.ietf.org/html/rfc6550).

IETF RFC 6554, An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL) (http://tools.ietf.org/html/rfc6554).

IETF RFC 6763, DNS-Based Service Discovery (http://tools.ietf.org/html/rfc6763).

IETF RFC 6960, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP (https://tools.ietf.org/html/rfc6960).

IETF RFC 7230, Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing (https://tools.ietf.org/html/rfc7230).

IETF RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content (https://tools.ietf.org/html/rfc7231).

IETF RFC 7232, Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests (https://tools.ietf.org/html/rfc7232).

IETF RFC 7233, Hypertext Transfer Protocol (HTTP/1.1): Range Requests (https://tools.ietf.org/html/rfc7233).

IETF RFC 7234, Hypertext Transfer Protocol (HTTP/1.1): Caching (https://tools.ietf.org/html/rfc7234).

IETF RFC 7235, Hypertext Transfer Protocol (HTTP/1.1): Authentication (https://tools.ietf.org/html/rfc7235).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

IETF RFC 7251, AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS (http://tools.ietf.org/html/rfc7251).

Internet Assigned Numbers Authority (IANA), Private Enterprise Number (PEN) Request.[3]

ISO 4217, Codes for the representation of currencies.[4]

ITU-T Recommendation X.680 (08/2015), Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation.[5]

NIST Special Publication (SP) 800-57, Recommendation for Key Management.[6]

NIST Special Publication (SP) 800-131A, Transition: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths.

W3C, Efficient XML Interchange (EXI) Format 1.0 (http://www.w3.org/TR/exi/).[7]

W3C, Extensible Markup Language (XML) (http://www.w3.org/TR/REC-xml/).

W3C, Web Application Description Language (WADL) (http://www.w3.org/Submission/wadl/).

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[8]

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 certificate authority (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.

[3] Available online: http://pen.iana.org/pen/PenApplication.page.

[4] ISO publications are available from the International Organization for Standardization (http://www.iso.org/) and the American National Standards Institute (http://www.ansi.org/).

[5] ITU-T publications are available from the International Telecommunications Union (http://www.itu.int/).

[6] NIST publications are available from the National Institute of Standards and Technology (http://www.nist.gov/).

[7] W3C publications are available from the World Wide Web Consortium (https://www.w3.org/).

[8] *IEEE Standards Dictionary Online* subscription is available at: http://dictionary.ieee.org.

Copyright © 2018 IEEE. All rights reserved.

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 IEEE 2030.5 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 IEEE 2030.5 features (e.g., metering, demand response, and load control).

Function Set Assignments: A logical addressing mechanism in IEEE Std 2030.5 that allows devices to be directed to use specific resources (e.g., to facilitate a device's participation in a program). Please see 8.6 for details.

function set instance: A single, top-level, instance of a function set (e.g., a single UsagePoint for the Metering function set, a single DemandResponseProgram for the Demand Response and Load Control function set).

host: This is the representation of a device in its application context. Typically represented by an Internet Protocol (IP) address or domain name.

intermediate CA: A certificate authority (CA) below the root CA that issues certificates to subordinate CAs.

issuing CA: A certificate authority (CA) that issues certificates to devices or code-signers.

manufacturer's CA (MCA): An intermediate certificate authority (CA) operated by a specific manufacturer for the purpose of issuing manufacturing issuing CAs for that manufacturer.

manufacturing issuing CA (MICA): An issuing certificate authority (CA) that issues certificates to devices during the manufacturing process.

manufacturing public key infastructure (PKI): The set of certificate authorities (CAs) that issue certificates to devices during the manufacturing process. The set includes the Smart Energy Root CA, Manufacturer's CAs, and Manufacturing Issuing CAs.

master resource identifier (mRID): An mRID is the global identifier attribute of an object. For further definition and formatting, see the schema (IEEE Std 2030.5 supplemental material).

node: This is the representation of a device in its network context, typically represented as an Internet Protocol (IP) address.

object identifier (OID): An OID consists of a node in a hierarchically-assigned namespace, formally defined using ITU-T's ASN.1 standard (ITU-T Recommendation X.680).

privacy-enhanced electronic mail (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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

public key infrastructure (PKI): 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: Uniform resource identifier (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 certificate authority (CA) whose certificate or public key is a trust anchor for any other certificates in a chain of trust.

root certificate: The root certificate authority'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 certificate authority (CA) for the IEEE 2030.5 manufacturing public key infastructure (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 IEEE 2030.5 hierarchy in the form of a self-signed certificate.

trusted root store: An integrity-protected location for storing root certificates.

3.2 Acronyms and abbreviations

This subclause provides a list of acronyms and abbreviations introduced in this document.

ACL	access control list
CA	certificate authority
CRL	certificate revocation list
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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

EMS	energy management system
ESI	energy services interface
EVSE	electric vehicle supply equipment
EXI	Efficient XML Interchange
HEMS	home energy management system
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
LFDI	long form device identifier
MCA	manufacturer’s certificate authority
MICA	manufacturing issuing certificate authority
OCSP	Online Certificate Status Protocol
OID	object identifier
PKI	public key infrastructure
REST	representational state transfer
SERCA	smart energy root certificate authority
SFDI	short form device identifier
TCP	Transmission Control Protocol
UOM	units of measure
URI	uniform resource identifier
WADL	Web Application Description Language
XML	Extensible Markup Language

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

4. Design pattern

4.1 Protocol flexibility

IEEE Std 2030.5 is designed to implement a REST architecture. It is built around the core actions of GET, HEAD, PUT, POST, and DELETE (as used in (Fielding [B3]), with the addition of a lightweight subscription mechanism as discussed in 8.7. Any application protocol that can implement a RESTful command set could likely be used with IEEE Std 2030.5, but Hypertext Transfer Protocol (HTTP) is a required baseline for interoperable IEEE 2030.5 implementations. HTTP utilizes Transmission Control Protocol (TCP) as its transport protocol. As a result, TCP manages the session providing delivery assurance and windowing.

IEEE 2030.5 servers and clients SHALL be compliant with IETF RFC 7230, IETF RFC 7231, IETF RFC 7232, IETF RFC 7233, IETF RFC 7234, and IETF RFC 7235.

4.2 General rules/best practices

This standard shall not make distinctions between servers, clients, or devices when defining interfaces and uniform resource identifiers (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 default polling rate for a given function set is specified at the top-level resource of that function set in the pollRate attribute.

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 Hypertext Transfer Protocol Secure (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 standard. 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 standard are recommendations. Thus, clients

Copyright © 2018 IEEE. All rights reserved.

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.

4.3 WADL

The IEEE 2030.5 RESTful interface is defined using the Web Application Description Language (WADL) (IEEE Std 2030.5 supplemental material).

IEEE 2030.5 devices SHALL conform to the interface specifications contained in the WADL as follows:

a) Devices SHALL conform to the WADL specification as per W3C WADL.

b) Devices SHALL conform to the WADL definition in the supplemental material of IEEE Std 2030.5. By implication, all resource representations SHALL validate per the schema (IEEE Std 2030.5 supplemental material) within the standardized IEEE 2030.5 XML namespace (`urn:ieee:std:2030.5:ns`).

c) Compliance (MODE) designations are interpreted as follows:

1) Mandatory ("M"): Devices MUST implement and conform.

2) Optional ("O"): Devices MAY implement, and if implemented, MUST conform.

3) Discouraged ("D"): Devices SHOULD NOT implement, but if implemented, MUST conform.

4) Error ("E"): Devices MUST return one of the specified response status codes (e.g., 400 Bad Request or 405 – Method Not Allowed).

4.4 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

4.5 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:

a) URI elements are at most four characters, but recognizable to a knowledgeable engineer. Element names as short as one character are acceptable provided their meaning is clear.

b) URI elements are constructed of consonants only, unless inclusion of a vowel adds clarity, such as a leading vowel or well-known abbreviation.

c) URI elements are in all lower case.

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

4.6 List resources

4.6.1 Introduction

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

The following attributes are defined for list resources:

— `all` is 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` is 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 created 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 standard 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.

Copyright © 2018 IEEE. All rights reserved.

4.6.2 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:

```
{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 "4294967295." If this query string parameter is not specified, the default start value SHALL be "0."
- `a` ("after") is used to indicate that only items whose primary key occurs after the given date/time parameter should be included in the query result list. This query string parameter is only applied to list resources that are ordered using a time-based primary key. The parameter SHALL be ignored if the primary key is not time-based. The format of the parameter SHALL be a 64-bit decimal number with identical semantics as that of the TimeType (see subclause 12.2 and the XML XSD in the supplemental material of IEEE Std 2030.5).
- `l` ("limit") is used to set the maximum number of list items (up to 4 294 967 295) to be included in the query result list. The value is specified in decimal. If this query string parameter is not specified, the default limit SHALL be "1." Servers MAY return a result list smaller than that specified by the client-provided limit.

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

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

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

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

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 third 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="7" 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>
    <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>
```

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:

```
<MyTypeList href="http://host1/the/list" all="7" results="1">
    <MyType href="http://host1/instance/of/type/red">
        <timeStamp>100</timeStamp>
    </MyType>
</MyTypeList>
```

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

```
<MyTypeList href="http://host1/the/list" all="7" results="5">
    <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>
```


Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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

```
<MyTypeList href="http://host1/the/list" all="7" results="1">
    <MyType href="http://host6/instance/of/type/white">
        <timeStamp>600</timeStamp>
    </MyType>
</MyTypeList>
```

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

```
<MyTypeList href="http://host1/the/list" all="7" results="2">
    <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?s=12&l=2` will return a list representation containing zero items:

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

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

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

```
<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:

```
<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:

```
<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:

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


Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

4.7 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 standard 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, for example, NotificationList, SHALL include an `xsi:type` attribute. In this case, the XML Schema Instance Namespace must also be declared. See Clause 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.

5. Application support

5.1 Overview

This clause details the standards-based application transport protocol and other lower layer requirements required for interoperability of IEEE 2030.5 devices.

The application support layer provides the following services:

- RESTful HTTP/1.1 as the application data exchange semantics, as described in IETF RFC 7231 and (Fielding [B3])
- XML (W3C XML) and/or EXI (W3C EXI) encoding as the data payload of the RESTful operations
- Transport authentication and encryption using HTTP over TLS (IETF RFC 2818 and IETF RFC 5246)

5.2 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. IETF RFC 7230 states that:

“HTTP messaging is independent of the underlying transport- or session-layer connection protocol(s). HTTP only presumes a reliable transport with in-order delivery of requests and the corresponding in-order delivery of responses. The mapping of HTTP request and response structures onto the data units of an underlying transport protocol is outside the scope of this specification.”

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 (IETF RFC 793).

5.3 URI encoding

The address of the resources presented by an IEEE 2030.5 host will use the standard URI syntax specific to HTTP/1.1 (i.e., `http://`) as per IETF RFC 1630 and IETF RFC 3986.

Given the constrained nature of many IEEE 2030.5 hosts, it is critical to devise the URL namespace scheme (hierarchy) that is both descriptive and compact. For more information, see 4.5.

URIs generated by IEEE 2030.5 hosts SHOULD use only xmDNS domain names or Internet Protocol (IP) addresses in fully qualified URIs, as IEEE 2030.5 devices may not have access to name servers.

5.4 HTTP headers

5.4.1 Introduction

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 IEEE 2030.5 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.

IEEE 2030.5 provides a set of *mandatory, optional,* and *discouraged* recommendations for each HTTP header. IEEE 2030.5 implementations SHOULD employ only mandatory HTTP headers, minimize the use of optional HTTP headers, and avoid the use of discouraged headers.

5.4.2 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.

Table 1 summarizes the recommended use of HTTP/1.1 headers in IEEE Std 2030.5. This table is provided as informative guidance, particularly to suit constrained networks and devices, and IETF RFC 7230, IETF RFC 7231, IETF RFC 7232, IETF RFC 7233, IETF RFC 7234, and IETF RFC 7235 are the authoritative references for compliance.

Copyright © 2018 IEEE. All rights reserved.

Table 1—HTTP headers

Header	Used in message type		IETF RFC required/optional	IEEE 2030.5 use
Accept	Request		Optional	Mandatory
Accept-Charset	Request		Optional	Discouraged
Accept-Encoding	Request		Optional	Discouraged
Accept-Language	Request		Optional	Discouraged
Accept-Ranges	Request	Response	Optional	Discouraged
Age		Response	Optional (required for a cache)	Discouraged
Allow		Response	Required	Mandatory
Authorization	Request	Response	Optional	Discouraged
Cache-Control	Request	Response	Optional	Discouraged
Connection	Request		Optional (required in some situations [e.g., HTTP/1.1 applications that do not support persistent connections])	Optional (mandatory in some situations)
Content-Encoding		Response	Optional (required when an encoding is applied)	Discouraged
Content-Language		Response	Optional	Discouraged
Content-Length	Request	Response	Optional (required in many situations, see Section 3.3.2 of IETF RFC 7230)	Optional (required in many situations, see Section 3.3.2 of IETF RFC 7230)
Content-Location		Response	Optional	Discouraged
Content-MD5		Response	Optional	Discouraged
Content-Range		Response	Optional	Optional (see 9.8.2.3)
Content-Type	Request	Response	Required	Mandatory
Cookies	Request		Optional	Discouraged
Date	Request	Response	Mandatory	Mandatory
ETag		Response	Optional	Optional (see 9.8.2.3)
Expect	Request		Optional	Discouraged
Expires		Response	Optional	Discouraged
From	Request		Optional	Discouraged
Host	Request		Required	Mandatory
If-Match	Request		Optional	Discouraged
If-Modified-Since	Request		Optional	Discouraged
If None-Match	Request		Optional	Discouraged
If-Range	Request		Optional	Discouraged
If-Unmodified-Since	Request		Optional	Discouraged
Last-Modified		Response	Optional	Discouraged
Location		Response	Optional	Mandatory in many situations (e.g., POST responses)
Max-Forwards	Request		Optional	Discouraged
Pragma	Request	Response	Optional	Discouraged
Proxy-Authenticate		Response	Optional	Discouraged
Proxy-Authorization	Request		Optional	Discouraged
Range	Request		Optional	Optional (see 9.8.2.3)
Referer	Request		Optional	Discouraged
Retry-After		Response	Optional	Discouraged
Server		Response	Optional	Discouraged
Set-Cookie		Response	Optional	Discouraged
TE	Request		Discouraged	Discouraged
Trailer		Response	Discouraged	Discouraged
Transfer-Encoding		Response	Optional	Discouraged
Upgrade	Request		Optional	Discouraged
User-Agent	Request		Optional	Discouraged
Vary		Response	Discouraged	Discouraged
Via	Request	Response		Discouraged
Warning	Request	Response	Discouraged	Discouraged
WWW-Authenticate		Response	Optional	Discouraged

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

5.5 HTTP response codes

5.5.1 Introduction

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 subclause attempts to highlight HTTP response codes that are felt to be more important or that need special attention from developers. However, IEEE 2030.5 clients may encounter various HTTP response codes and, all use of, and response to HTTP response codes SHALL be standard and RFC compliant.

5.5.2 Common responses

5.5.2.1 Introduction

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

5.5.2.2 1xx (Informational)

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

IETF RFC 7231 states, “A client MUST be able to parse one or more 1xx responses received prior to a final response, even if the client does not expect one. A user agent MAY ignore unexpected 1xx responses.”

5.5.2.3 200 ("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”]).

5.5.2.4 201 ("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.

IETF RFC 7231 states, “If the target resource does not have a current representation and the PUT successfully creates one, then the origin server MUST inform the user agent by sending a 201 (Created) response.”

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

5.5.2.5 204 ("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).

5.5.2.6 206 ("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 IETF RFC 7233 requires the Content-Range and Date headers MUST be present in the response.

5.5.2.7 301 ("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.

5.5.2.8 302 ("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 URIs requested as HTTP to HTTPS.

5.5.2.9 400 ("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.

IETF RFC 7230 states, “A server MUST respond with a 400 (Bad Request) status code to any HTTP/1.1 request message that lacks a Host header field and to any request message that contains more than one Host header field or a Host header field with an invalid field-value.”

Copyright © 2018 IEEE. All rights reserved.

5.5.2.10 401 ("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.

IETF RFC 7235 states, “The server generating a 401 response MUST send a WWW-Authenticate header field (Section 4.1) containing at least one challenge applicable to the target resource.”

5.5.2.11 404 ("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.

5.5.2.12 405 ("Method Not Allowed")

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

IETF RFC 7231 states, “The origin server MUST generate an Allow header field in a 405 response containing a list of the target resource’s currently supported methods.”

5.5.2.13 406 ("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.

5.5.2.14 413 ("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.

5.5.2.15 416 ("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.

5.5.2.16 417 ("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.

5.5.2.17 500 ("Internal Server Error")

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

5.5.2.18 501 ("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.

5.5.2.19 503 ("Service Unavailable")

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

5.5.3 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 an unrecognized status code as being equivalent to the x00 status code of that class, with the exception that a recipient MUST NOT cache a response with an unrecognized status code” (IETF RFC 7231).

5.6 Application payload syntax

5.6.1 Introduction

Application payload message encoding using both XML (W3C XML) and EXI (W3C EXI) SHALL be supported by all servers. Application payload message encoding using either XML (W3C XML) or EXI (W3C EXI) SHALL be supported by all clients. Certain function sets may require additional encodings and, if so, are specified in their respective sections.

5.6.2 XML encoding

The XML declaration is optional as per W3C XML and SHOULD NOT be included in IEEE 2030.5 transactions, to reduce packet sizes. The XML version used SHALL be 1.0. For XML payloads, the encoding SHALL be UTF-8.

5.6.3 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 (W3C EXI) defaults and SHOULD NOT be specified explicitly.

- Non-strict schema-informed grammar with the schema (IEEE Std 2030.5 supplemental material)
- 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)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

— schemaId is “S1” (two bytes: 0x53, 0x31, without quotes). This schemaId corresponds to the normative schema of IEEE Std 2030.5-2018 (supplemental material).

— 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:

```
<header xmlns="http://www.w3.org/2009/exi">
    <common><schemaId>S1</schemaId></common>
</header>
```

5.7 Content negotiation

5.7.1 General

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

5.7.2 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 IETF RFC 7231. The extensibility level specified using an HTTP Accept header supersedes an extensibility declaration discovered during resource discovery (see 7.4).

The extensibility level defines the base schema and its capability for arbitrary extension. The extensibility level is one of “-S1” or “+S1”. The S1 indicates the base schema version: IEEE Std 2030.5-2018. “-S1” indicates the node does not accept arbitrary tags that are not defined in the base schema, and “+S1” indicates it accepts arbitrary tags. A node with “-S1” 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 “-S1” 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 “1”). As extensions of IEEE 2030.5 schemas are intended to be backward compatible, a node

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

that declares schemaId “S[i]” is intended to be compatible with all versions between “S1” and “S[i]”. However, an “S1” implementation is not backward compatible with an “S0” (IEEE 2030.5-2013) implementation.

For example:

```
Accept: application/sep-exi; level=-S1
```

indicates that the client wishes to receive content encoded using EXI where the base schema of the client is IEEE 2030.5-2018 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.

6. Security

6.1 Introduction

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 clause describes the security features that are provided at the application layer and that are REQUIRED for use over all networks.

Securing transactions between clients and servers is based on using HTTP over TLS (IETF RFC 2818) (also known as HTTPS) using TLS version 1.2 (IETF 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.

6.2 Security attributes

6.2.1 Introduction

In this subclause, we define some abstract data structures for managing registration (see 6.9) and access control. How this functionality is accomplished is left to the implementer. No access to these data structures is defined in this standard.

6.2.2 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.

Copyright © 2018 IEEE. All rights reserved.

Table 2—Local registration attributes

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

Table 3—Local registration descriptor entry

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

6.2.3 Access control list (ACL) attributes

6.2.3.1 Introduction

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 subclause.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 4—ACL attributes

Attribute	Identifier	Type	Range	Description	Default
aclDefaultAccess	0x02	Access descriptor		Default access to resource	-
aclSpecificID	0x03	List		A list of specificIDDescriptors for each specific client access to resource	-
aclSpecificIDEntries	0x04	Integer	Implementation specific	Number of entries in *aclSpecificID*	0

Table 5—AccessDescriptorEntry

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

Table 6—SpecificIDDescriptor entry

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

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 standard. Access control that is more granular than a resource is out of scope for this standard. The mechanisms for authentication and the binding of that authentication to a specified identity (such as an IP address) are specified in 6.5 and 6.6.

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

Copyright © 2018 IEEE. All rights reserved.

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 standard and is governed by overall security policy. Default settings for ACLs for particular function sets are described in 6.8.

6.2.3.2 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.

6.2.3.3 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).

6.2.3.4 Access authorization

6.2.3.4.1 Introduction

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 5 and Table 6.

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

6.2.3.4.2 Method attribute

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

— GET

— PUT

— POST

— DELETE

— HEAD

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

6.2.3.4.3 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 standard 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
```

6.2.3.4.4 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 6.11.7.2).

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 (any device type).

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.

6.2.3.5 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
```


Copyright © 2018 IEEE. All rights reserved.

or

```
HTTP/1.1 405 Method Not Allowed
```

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.

6.2.3.6 ACL examples

6.2.3.6.1 Introduction

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

6.2.3.6.2 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 7—Example ACL for EndDeviceList

Attribute	Comment
aclDefaultAccess	Method: 0x01 (GET only) AuthType: 0x1 (no authentication) DeviceType: 0 (any device type)
aclSpecificID	Empty
aclSpecificIDEntries	0

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

6.2.3.6.3 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 shown in Table 8.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 8—Example ACL entry for EndDevice at point of registration

Attribute	Comment
aclDefaultAccess	Method: 0x00 (no default access) AuthType: 0x1 (no authentication) DeviceType: 0 (any device type)
aclSpecificID	Empty
aclSpecificIDEntries	0

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, as shown in Table 9.

Table 9—Example local registration entry for registering device

Name	Type	Range	Description	Default
PIN	Integer	0 to 999 999	6-digit PIN (5 plus check digit) of registering device	-
SFDI	SFDI	0 to 68 719 476 735, $2^{36}-1$	SFDI of registering device	-
DeviceType	Integer	0 to 3	Registering device type	0
HardwareModuleName	String	-	Optional additional hardware module information	-

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 as shown in Table 10 and Table 11.

Table 10—Example ACL entry for EndDevice at point of client access

Attribute	Comment
aclDefaultAccess	Method: 0x00 (no default access) AuthType: 0x1 (no authentication) DeviceType: 0 (Any device type)
aclSpecificID	One SpecificIDDescriptor for client
aclSpecificIDEntries	1

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 11 —Example SpecificIDDescriptor for EndDevice at point of client access

Name	Type	Range	Description	Default
Access	AccessDescriptor	-	Access levels required for client: Method: 0xF (GET, PUT, POST, DELETE) AuthType: 0x8 (device certificate) DeviceType: (as appropriate for device)	-
IPAddr	IPAddr	-	IP address of client	-
Port	Integer	0x0000 to 0xffff	Port of client	0

6.3 Device credentials

6.3.1 Introduction

There are three credentials per device:

— Short form device identifier (SFDI)

— Long form device identifier (LFDI)

— PIN

6.3.2 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

6.3.3 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 6.3.2, 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 home area network (HAN) or site domain. It should not

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

6.3.4 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 6.3.2, 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.

6.3.5 6-digit PIN code

The SFDI 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.

6.3.6 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.

6.4 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 6.9) with a utility or third party service provider MAY also be needed to provide explicit device and user authorization at the application layer.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 (IETF RFC 2818) (also known as HTTPS) using TLS version 1.2 (IETF 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 (IETF RFC 7230).

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 6.5. 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 that the client is unauthenticated will also be compared with the ACL (see 6.2.3.4.3).

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.

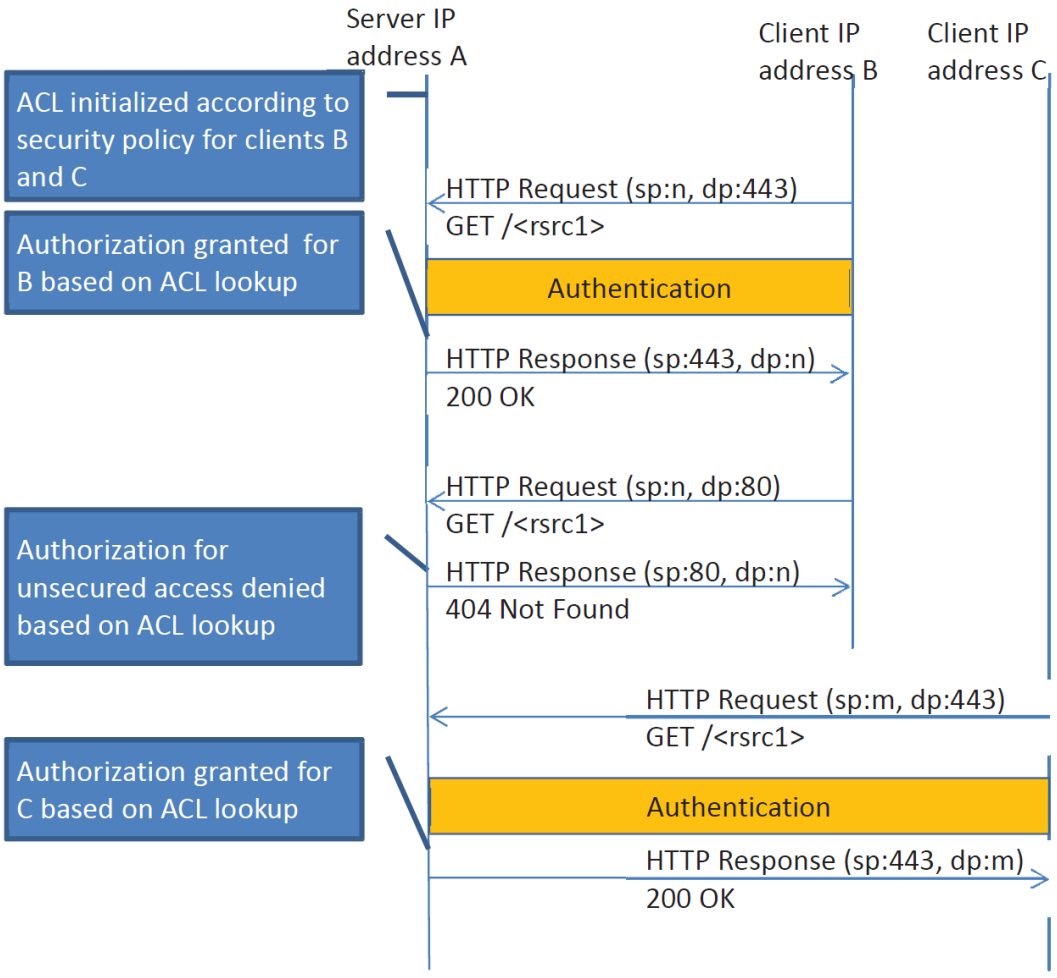


Figure 1—Example device authentication procedure using HTTPS port 443

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

6.5 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 standard.

The use of TLS (IETF 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 (IETF 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 (IETF 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).

6.6 Resource access authorization

Pre-authorization for resources is normally set when the client registers with the host as described in 6.9. 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.

6.7 Cipher suites

All devices SHALL support the TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite (IETF RFC 7251). The ECC cipher suite SHALL use elliptic curve secp256r1.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

— 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.

Devices MAY support additional cipher suites; however, any additional cipher suites SHOULD provide a cryptographic strength at least equivalent to the mandatory cipher suite.

6.8 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 standard. 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 12 reflects the functionsImplemented attribute in DeviceInformation.

Table 12—Attribute values for default security policy

Function set	*aclDefaultAccess* AuthType	Device certificate needed	Registered device
Device capability	0xf	No	No
Self device resource	0xc	No	Yes
End device resource	0xc	No	Yes
Function set assignments	0x8	Yes	Yes
Subscription/Notification mechanism	0x8	Yes	Yes
Response	0x8	Yes	Yes
Time	0x8	Yes	Yes
Device information	0x8	Yes	Yes
Power status	0x8	Yes	Yes
Network status	0x8	Yes	Yes
Log event	0x8	Yes	Yes
Configuration resource	0x8	Yes	Yes
Software download	0x8	Yes	Yes
DRLC	0x8	Yes	Yes
Metering	0x8	Yes	Yes
Pricing	0xc	No	Yes
Messaging	0xc	No	Yes
Billing	0x8	Yes	Yes
Prepayment	0x8	Yes	Yes
Flow reservation	0x8	Yes	Yes
DER control	0x8	Yes	Yes

The *aclDefaultAccess* attribute Method value SHOULD match the Allowed Methods for each resource enumerated in the IEEE 2030.5 WADL (IEEE Std 2030.5 supplemental material). The Method value MUST contain GET (0x01). The *aclDefaultAccess* attribute DeviceType value should be “any device type” (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

Copyright © 2018 IEEE. All rights reserved.

any IEEE 2030.5 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.

6.9 Registration

6.9.1 Introduction

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 subclause 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 energy services interface (ESI) associated with the utility, premises owner, or third party service provider that is trusted to perform registration.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

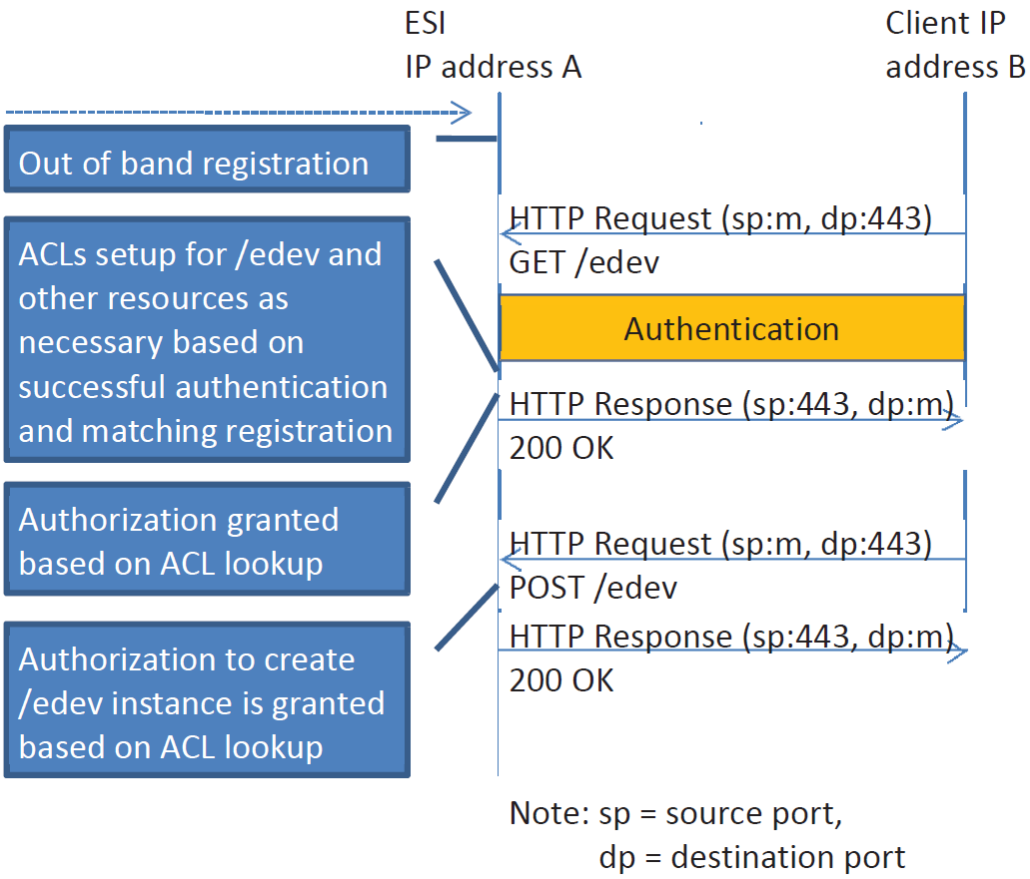


Figure 2—Device authentication with registration procedure examples using HTTPS

6.9.2 EndDeviceList

Clients SHALL locate HAN services by performing DNS service discovery (DNS-SD) queries to the HAN; see Clause 7 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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:

a) The EndDevice resource's server SHALL listen on the TCP port associated with HTTPS and follow the procedure described in (IETF RFC 5246) when a client attempts to access the EndDevice resource.

b) 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.

c) 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.

d) 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.

6.10 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 shown in Table 13.

Table 13—Security LogEvents

LogEvent name	LogEvent code	LogEvent description
SEC_TLS_ALERT	0x00	SHOULD be issued when a TLS Alert is generated. The logEventID SHALL be set to the TLS Alert value (IETF RFC 5246).
SEC_REGISTRATION_MISS	0x01	SHOULD be issued when a received certificate does not have a corresponding SFDI entry in the registration list.
SEC_ACL_ACCESS_FAILED	0x02	SHOULD be issued when access to a resource fails due to failing access control criteria described in 6.2.3.

6.11 Certificate management

6.11.1 Introduction

This standard defines a public key infrastructure (PKI) in the IEEE 2030.5 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 IEEE 2030.5 devices implementing IEEE 2030.5 applications over TLS. It is expected that the market will implement one or more Manufacturing PKIs in accordance with the requirements set forth in this standard.

In addition, this document describes a few other types of certificates that IEEE 2030.5 applications may encounter depending on which services they support—see 6.11.8.4. While native IEEE 2030.5 devices are required to understand, support, and process Manufacturing PKI certificates, support of the additional

Copyright © 2018 IEEE. All rights reserved.

certificates is optional and generally targeted at specific classes of devices (e.g., energy services interface [ESI], web portal).

There are six classes of certificates that may be active in an IEEE 2030.5 system, depending on configuration and use, as follows:

- Device certificates: Issued under the Manufacturing PKI during manufacturing to purpose-built (aka “native”) IEEE 2030.5 certified devices for operational purposes.
- Device test certificates: Issued under the Manufacturing PKI during manufacturing to purpose-built (aka “native”) IEEE 2030.5 certified devices for test purposes.
- Additional certificates for IEEE 2030.5 devices: One or more OPTIONAL TLS server certificates issued by non-IEEE 2030.5 CAs to IEEE 2030.5 devices such as ESIs for use in addition to the device certificate.
- Generic client certificate for non-native entities: A TLS client certificate issued by a generic (non-IEEE 2030.5) certificate authority to a non-native entity.
- Generic server certificate for non-native entities: A TLS server certificate issued by a generic (non-IEEE 2030.5) 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.

6.11.2 Certificate usage—authentication versus 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 14 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 “IEEE 2030.5 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) (IETF RFC 6960) as an additional mechanism to determine if a certificate has been revoked. OCSP may only be used to verify non-IEEE 2030.5 certificates.

Table 15 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 14—TLS authentication matrix

Authentication		Server		
		Native IEEE 2030.5 application	Generic server	Self-signed
Client	Native IEEE 2030.5 application	IEEE 2030.5 Cert - Indef	Optional OCSP	Not allowed
	Generic client	Optional OCSP	Not specified here	Not specified here
	Self-signed	Signature validation	Not specified here	Not specified here

Table 14 should be read as describing the authentication on the offered client certificate by the server. For example, Generic client/Native IEEE 2030.5 application has an "Optional OCSP" entry meaning the IEEE 2030.5 Application can OPTIONALLY use OCSP to check the validity of the Generic Client certificate in addition to its normal certificate validation process.

"IEEE 2030.5 Cert - Indef" means that if the IEEE 2030.5 Manufacturing PKI certificate validly chains to the IEEE 2030.5 root it is considered valid—neither OCSP nor CRLs are used or issued by CAs within the Manufacturing PKI hierarchy.

Table 15—IEEE 2030.5 authorization matrix

	Native IEEE 2030.5 application	Generic server	Self-signed
Native IEEE 2030.5 application	ACL	ACL or public resources (server specific)	Not allowed
Generic client	ACL or public resources	Not specified here	Not specified here
Self-signed	ACL or public resources	Not specified here	Not specified here

The above table should be read as describing the authorization mechanism used by the server with respect to the offered client credential. For example, the Generic Client/Native IEEE 2030.5 application entry of "ACL" means that either there is a specific ACL for the generic client credential that permits access to specific data, or the absence of such ACL allows that generic client access only to public resources.

6.11.3 Manufacturing PKI

6.11.3.1 Introduction

This subclause covers only those certificates issued under the auspices of the Manufacturing PKI. It specifically excludes the certificates described in 6.11.8.4 as "Other Certificates."

Any IEEE 2030.5 Manufacturing PKI SHOULD be a hierarchy with a depth of 2, 3, or 4 levels. At the top level, Manufacturing PKI hierarchy SHOULD have one SERCA. 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. Private key material for a SERCA SHOULD 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.

A SERCA MAY issue device certificates on behalf of one or more manufacturers.

The Manufacturing PKI hierarchy MAY include one IEEE 2030.5 MCA. One or more MCAs SHOULD be out-sourced to an IEEE 2030.5 vendor, specifically for the issuance of vendor-specific MICAs. An IEEE

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

2030.5 vendor MAY contract with a commercial CA to host the IEEE 2030.5 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 IEEE 2030.5 MICA. One or more MICAs SHOULD be out-sourced to an IEEE 2030.5 vendor, specifically for the production of IEEE 2030.5 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

6.11.3.2 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 on 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 destroy[9] 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 6.11.8.3.1 and 6.11.8.3.2 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.

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

6.11.3.3 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.

6.11.3.4 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).

6.11.4 General certificate format

6.11.4.1 RFC 5280 compliance

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

6.11.4.2 IEEE 802.1AR compliance

Device certificates and device test certificates in the Manufacturing PKI SHALL be compliant with IEEE Std 802.1AR™ with the following exceptions:

- Device certificates and device test certificates take the general form of an iDevID certificate as defined in IEEE Std 802.1AR.
- Differing from IEEE Std 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.[10] 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 IETF RFC 4108. Per IETF RFC 5280, this extension MUST be marked critical when the SubjectName field is empty. The hwType field of HardwareModule name is assigned by the IEEE 2030.5 manufacturer and SHOULD be different for each different type of manufactured device.

6.11.5 General restrictions and conditions

In addition, IEEE 2030.5 certificates have the following restrictions:

- All IEEE 2030.5 certificates are X.509 v3 certificates as defined in IETF RFC 5280.
- The only permitted public key type for IEEE 2030.5 certificates in the Manufacturing PKI is an Elliptic Curve (EC) public key on the NIST P-256 curve. (See 6.11.8.4.4 and 6.11.8.4.5 for details on the use of RSA Public Keys and certificates.)

[10] 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 Std 802.1AR, we constrain the identity expression to just the SubjectAlternativeName format described in IEEE Std 802.1AR.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

— The signature method for signatures formed by EC P-256 private keys MUST be SHA256withECDSA.

— 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 IETF RFC 5480 Section 2.2 for details on the uncompressed form.

— Per IETF 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.[11]

— Per IETF 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 6.11.8.3.3 and 6.11.8.3.4, the SubjectName MUST be non-empty.

6.11.6 Extensions

IEEE 2030.5 certificate extensions have the following restrictions:

— 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.

— In the Manufacturing PKI hierarchy, each device certificate and the CAs that make up the device certificate's path contain one or more device type identifiers encoded in the certificatePolicy extension in the PolicyInformation: policyIdentifier field. These policyIdentifier Object Identifiers (OIDs) (IETF RFC 5280) are taken from those OIDs defined under the deviceType arc of the IEEE 2030.5 OID tree. See 6.11.7.2 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 IETF RFC 5280 Section 4.1.2.5 and a Valid:notAfter field consisting of the GeneralizedTime value `99991231235959Z` (see IETF 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 IETF 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 IETF RFC 5280. In both cases, the extension MUST be marked non-critical.

— IEEE 2030.5 devices MUST be able to follow a chain where the key identifier was not generated in compliance with this subclause, 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.

[11] 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 $\cong$ 600 000 certs, 10 octets $\cong$ 155 $\times 10^{6}$ certs, 11 octets $\cong$ 2.5 $\times 10^{9}$ certs signed without serial number collision.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

6.11.7 Additional ASN1 definitions

6.11.7.1 Introduction

The IEEE 2030.5 object identifier arc has been allocated by the IANA as follows:

```
ieee2030.5 OBJECT IDENTIFIER ::= { iso(1) identified-organizations(3) dod(6) internet(1)
private(4) enterprise(1) 40732 }
```

6.11.7.2 IEEE 2030.5 device type assignments

The deviceType is included in the CertificatePolicy extension of the device certificate and its issuing chain of CA certificates. It may be used for authorization purposes as described in 6.2.3.4.4.

```
deviceType OBJECT IDENTIFER ::= { ieee2030.5 1 }

id-IEEE 2030.5-dev-genericIEEE 2030.5Device OBJECT IDENTIFIER ::= { deviceType 1 }
-- used for most devices

id-IEEE 2030.5-dev-mobile OBJECT IDENTIFIER ::= { deviceType 2 }
-- used in addition to genericIEEE 2030.5Device to identify "mobile" IEEE 2030.5
-- entities (may be homed to multiple ESI domains)

id-IEEE 2030.5-dev-postManufactureIEEE 2030.5 OBJECT IDENTIFIER ::= { deviceType 3 }
-- used in device certs issued post-manufacture
```

6.11.7.3 IEEE 2030.5 policy assignments

One or more of IEEE 2030.5 Policy OIDS MAY be included in the device certificate and its issuing chain of CA certificates.

```
IEEE 2030.5 Policy OBJECT IDENTIFIER ::== { ieee2030.5 2 }

id-IEEE 2030.5-po-device-auth-test OBJECT IDENTIFIER ::= { IEEE 2030.5 Policy 1 }
-- MUST be included in test certificates

id-IEEE 2030.5-po-selfsigned-client OBJECT IDENTIFIER ::= { IEEE 2030.5 Policy 2 }
-- MUST be included in IEEE 2030.5 self-signed certificates

id-IEEE 2030.5-po-service-provider OBJECT IDENTIFIER ::= { IEEE 2030.5 Policy 3 }
-- MUST be included in commercial certificates issued to
-- service providers for IEEE 2030.5 purposes.

Id-IEEE 2030.5-po-bulk-cert OBJECT IDENTIFIER ::= { IEEE 2030.5 Policy 4 }
-- MUST be included in bulk-issued certificates (e.g.,
-- where the private key is generated off the device by the
-- issuing CA
```

6.11.7.4 HardwareModuleName

Excerpted from IETF 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 }
```


Copyright © 2018 IEEE. All rights reserved.

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 IEEE 2030.5 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. For 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
```

6.11.8 Certificate profiles

6.11.8.1 Introduction

The certificates listed here have the normal (IETF RFC 5280) format and the descriptions for all the fields listed in the subsequent subclauses are described in IETF 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 IETF RFC 5280 requires it to be identical to the Subject Name of the issuing certificate.

By specification, IETF 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).

IEEE 2030.5 devices MUST accept unexpected (not listed in this profile) certificate extensions and MUST silently ignore non-critical unrecognized certificate extensions. Per IETF RFC 5280, devices MUST reject any certificate containing unrecognized critical certificate extensions.

6.11.8.2 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=IEEE 2030.5 Root, serialNumber=<n>
- Subject Name: O=Smart Energy, CN=IEEE 2030.5 Root, serialNumber=<n>
- Extensions
  - certificatePolicy: Critical; 1:anyPolicy

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

- keyUsage: Critical; keyCertSign, crlSign
- basicConstraints: Critical; cA=true, pathLen absent (unlimited)
- subjectKeyIdentifer: see 6.11.6

NOTE—The root certificate is primarily a container for a Trust Anchor.[12]

6.11.8.3 Manufacturing hierarchy certificates

6.11.8.3.1 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: C=<country>, O=<Manufacturing Org>, CN=IEEE 2030.5 MCA, serialNumber=<num>
- Extensions:
  - certificatePolicy: critical; at least one IEEE 2030.5 device type Identifier OID as a policyIdentifier
  - keyUsage: critical; keyCertSign
  - basicConstraints: critical; cA=true, pathLen=1
  - subjectKeyIdentifier: see 6.11.6
  - authorityKeyIdentifier: see 6.11.6

6.11.8.3.2 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: C=<country>, O=<Manufacturing Org>, CN=IEEE 2030.5 MICA, serialNumber=<num>
- Extensions:
  - certificatePolicy: critical; at least one IEEE 2030.5 device type Identifier OID as a policyIdentifier
  - keyUsage: critical; keyCertSign
  - basicConstraints: critical; cA=true, pathLen=0
  - subjectKeyIdentifier: see 6.11.6
  - authorityKeyIdentifier: see 6.11.6

[12] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

6.11.8.3.3 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 6.11.4, the certificate is compliant with IEEE Std 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 IEEE 2030.5 device type identifier OID as a policyIdentifier
  - subjectAlternativeName: critical; one GeneralName of type OtherName of hardwareModuleName (see 6.11.7.4 above for specific field values)
  - keyUsage: critical; one or more of keyAgreement, digitalSignature
  - authorityKeyIdentifier: see 6.11.6

6.11.8.3.4 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 6.11.4, the certificate is compliant with IEEE Std 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-IEEE 2030.5-po-device-auth-test as policyIdentifier in the certificatePolicy extension.

6.11.8.4 Other certificates

6.11.8.4.1 Introduction

There are a few other certificates that an IEEE 2030.5 device may see.

6.11.8.4.2 Additional certificates for IEEE 2030.5 native devices

6.11.8.4.2.1 General considerations

In addition to the certificates described in 6.11.8.3.3, 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-IEEE 2030.5 Manufacturing certificates incorporating RSA public keys. The purpose for doing so is to provide

Copyright © 2018 IEEE. All rights reserved.

backwards-compatible support to client devices, software, and systems that may not yet support Elliptic Curve.

Non-IEEE 2030.5 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.

6.11.8.4.2.2 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 SHA256withRSA. They SHOULD contain an expiration data not later than 31 December, 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 31 December 2020.

The IEEE 2030.5 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 standard 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.

6.11.8.4.2.3 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 an IEEE 2030.5 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-IEEE 2030.5 RSA certificate.
- If an IEEE 2030.5 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 an IEEE 2030.5 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.

6.11.8.4.3 Self-signed client certificate

Any application designed to talk to native IEEE 2030.5 products or applications implementing IEEE 2030.5 functions may create a credential for itself consisting of a self-signed (IETF RFC 5280) certificate.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

The application generates the key pair and binds the public key in a certificate. The credential privilege is installed in the IEEE 2030.5 device by either passing the certificate or a fingerprint of the certificate along with the credential privileges to the appropriate IEEE 2030.5 application using an enrolment protocol (not specified here) or other mechanism (e.g., web portal mechanism for moving from unsecure to secure).

An IEEE 2030.5 device or application acting in a client role SHALL reject a self-signed certificate if presented by the server. An IEEE 2030.5 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=<12-digit random hex string>
- Issuer Name: Identical to Subject Name
- Validity: notBefore: time of issue; notAfter: maximum of time of issue plus three years
- Subject Public Key and Signature: SHOULD be EC P-256 and SHA256withECDSA, MAY be RSA 2048 and SHA256withRSA
- Extensions:
  - keyUsage: critical; at least digitalSignature, others as appropriate
  - certificatePolicy: critical; at least one policyIdentifier:id-IEEE 2030.5-po-selfsigned-client

6.11.8.4.4 Generic client certificates for non-IEEE 2030.5 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 IEEE 2030.5 protocol. As with a self-signed certificate, the credential privilege is installed in the IEEE 2030.5 device or application by either passing the certificate or a fingerprint of the certificate along with the credential privileges. At a later date, IEEE 2030.5 may specify further uses for certificates not issued under the IEEE 2030.5 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 IEEE 2030.5 definition are those that do not contain a basicConstraints extension.

6.11.8.4.5 Generic server certificates for non-IEEE 2030.5 entities

IEEE 2030.5 devices or applications may need to connect to TLS servers secured by server certificates issued by global or local CAs not affiliated with IEEE 2030.5. 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

Copyright © 2018 IEEE. All rights reserved.

In addition, the IEEE 2030.5 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 IEEE 2030.5 devices to external devices is subject to further specification.

6.11.9 Device requirements

6.11.9.1 Private-key protection

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

6.11.9.2 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.

6.11.9.3 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).

6.11.9.4 Trusted certificate store

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.

6.11.10 Certificate verification

6.11.10.1 Introduction

IEEE 2030.5 devices and applications MUST follow the procedure defined in Section 6 of IETF RFC 5280, 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

6.11.10.2 Additional considerations for serving IEEE 2030.5 to non-native entities

An IEEE 2030.5 device MAY implement OCSP for the sole purpose of validating non-IEEE 2030.5 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, an IEEE 2030.5 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. An IEEE 2030.5 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-IEEE 2030.5 device or application may use CRLs or OCSP for the purpose of attempting to validate IEEE 2030.5 certificates.

6.11.11 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 6.3. These data may be used as appropriate to label a device or its packaging.

7. Discovery

7.1 Introduction

IEEE 2030.5 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 IEEE 2030.5 plus its underlying transport bindings (e.g., HTTP(S)/TCP/IP). DNS-based Service Discovery (DNS-SD) (IETF 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 IEEE 2030.5, DNS-SD (IETF 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) IETF 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 (IETF RFC 4291).

Extended Multicast DNS (xmDNS) (IETF Extended Multicast DNS Internet Draft) 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. IEEE 2030.5 SHALL use global addresses or Unique Local Addresses (IETF RFC 4193) in the source address of xmDNS requests and responses. Extended Multicast DNS (xmDNS) (IETF Extended Multicast DNS Internet Draft) is normative for IEEE Std 2030.5.

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.

If a device plans to communicate outside of the local network (i.e., over the Internet), it SHOULD support unicast DNS with DNS-SD. However, it should be noted that devices may not be routable to the global Internet or may not have access to a DNS name server. Further, not all devices will support name resolution via unicast DNS and thus unicast DNS should be used carefully. Devices SHOULD NOT use unicast DNS and DNS-SD unless they have been explicitly provided a target. If it is desired for a client to communicate to a server not present on the local network, there are multiple methods to inform the client of the server, including:

- Using FunctionSetAssignments on the local network to direct the client to a server not on the local network.
- Communicating a DNS name out of band (e.g., at manufacture, via a user interface) and then subsequently performing DNS-SD.
- Communicating a DNS name or IP address out of band (e.g., at manufacture, via a user interface), along with the necessary elements (e.g., the URI of the DeviceCapabilities resource).

7.2 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 17 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

Copyright © 2018 IEEE. All rights reserved.

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.

7.3 Service name

The Service Name used with IEEE 2030.5 DNS-SD SHALL be `smartenergy` and has been registered appropriately with the Internet Assigned Numbers Authority (IANA).

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 IEEE 2030.5 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).

7.4 TXT record

This subclause specifies the format of the TXT record to be used in conjunction with IEEE 2030.5 DNS-SD. IEEE 2030.5 SHALL use a single TXT record format. The following table describes the supported TXT record parameters for IEEE Std 2030.5.

Table 16—TXT record parameters

Key=Value	Example
txtvers={#}	txtvers=1
dcap={relative reference to DeviceCapabilities}	dcap=/dcap
path={relative reference to the function set} corresponding to the specified subtype name	path=/upt
https={port }	https=443
level={schema extensibility level indicator}	level=-S1

`txtvers` SHALL be the first key in the TXT record. For IEEE Std 2030.5-2018, 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 5.7). 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 "-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 server's schema level extensibility indicator.

7.5 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.

IEEE 2030.5 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 7.2 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 7.4.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 IEEE 2030.5 servers in the HAN is STRONGLY DISCOURAGED except for diagnostic purposes. Instead, devices SHOULD use subtype queries as described in this clause to enumerate IEEE 2030.5 function sets.

The TXT record of the SRV/TXT record pair referenced by a subtype PTR record SHALL conform to the definition given in 7.4 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 composed of a subtype string, followed by "_sub._smartenergy._tcp.site.". Subtype strings SHALL NOT begin with an underscore (see Table 17). For example, the subtype name for the meter Usage Point function set shall be composed as "upt._sub._smartenergy._tcp.site.". Other subtype names SHALL (except as noted below) be composed per the meter Usage Point example.

Table 17 lists the defined service subtype strings and their corresponding IEEE 2030.5 function sets.

Table 17—Service subtype strings and their corresponding IEEE 2030.5 function sets

Subtype	Device capabilities field name	IEEE 2030.5 function set
bill	CustomerAccountListLink	Billing
derp	DERProgramListLink	Distributed Energy Resources
dr	DemandResponseProgramListLink	Demand Response and Load Control
edev	EndDeviceListLink	End Device
file	FileListLink	File Download
msg	MessagingProgramListLink	Messaging
mup	MirrorUsagePointListLink	Metering Mirror
ppy	PrepaymentListLink	Prepayment
rsps	ResponseSetListLink	Response
sdev	SelfDeviceLink	Self Device
tm	TimeLink	Time
tp	TariffProfileListLink	Pricing
upt	UsagePointListLink	Metering

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 6.3.3), and followed by "_sub._smartenergy._tcp.site.". For example, a server having an EndDevice resource for the remote device with SFDI `222222222228` would register a subtype PTR record named "edev-222222222228._sub._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 EndDevice resource of the remote device.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

7.6 Discovery procedure

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

Link-layer joining is not covered in this standard. 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 subclauses in this document, therefore these may need to be read before this sequence is fully understood.

a) Use xmDNS/DNS-SD to locate the servers with the function sets of interest (See IETF Extended Multicast DNS Internet Draft, and IETF RFC 6763).

b) For each server do the following:

   1) Establish TLS session if required (see Clause 6).

   2) GET the Device Capabilities resource (See 8.3).

   3) Look for the desired function set in the Device Capabilities resource.

   4) If there is an entry in Device Capabilities it will contain a URI of the entry point for that function set.

   5) Alternately, use the `path` returned in the subtype query response (See 7.5).

c) 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 (IETF RFC 7231).

8. Support resources

8.1 Introduction

This clause defines resources and function sets that provide operational information to the end devices of an IEEE 2030.5 network or provide those end devices with services to manage and support their operation. The clause begins with a general description of how each subclause of this and the following two clauses are organized.

Copyright © 2018 IEEE. All rights reserved.

8.2 Resource section outlines

8.2.1 Introduction

This subclause gives the reader a basic understanding of the outline of each of the sections describing the resources of IEEE Std 2030.5. 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, IEEE Std 2030.5 (this document), the IEEE Std 2030.5 XML Schema, and the IEEE Std 2030.5 WADL described in IEEE Std 2030.5 supplemental material. All three need to be consulted to get a full understanding of IEEE Std 2030.5.

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.

8.2.2 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.

8.2.3 List ordering

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

Table 18—Function set list ordering

Resource name	Primary key	Secondary key	Tertiary key
Resource	Resource.elementA (descending or ascending)	Resource.elementC (descending or ascending)	N/A
Sub-Resource	Sub-Resource.elementD (descending or ascending)	Sub-Resource.elementB (descending or ascending)	N/A

8.2.4 Application guidelines/behavior

This subclause 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 the supplemental material of IEEE Std 2030.5. This subclause 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 subclause may also include application guidelines germane to sleepy HAN devices.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

8.2.5 LogEvents

This subclause 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 19.

Table 19—Example LogEvents

LogEvent name	LogEvent code	LogEvent description
LE_EXAMPLE_0	0x00	Normative text defining when the event should be generated
LE_EXAMPLE_1	0x01	Normative text defining when the event should be generated

8.3 Device Capabilities function set

8.3.1 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.

8.3.2 List ordering

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

8.3.3 Application guidelines/behavior

IEEE 2030.5 clients locate HAN services by performing DNS service discovery (DNS-SD) queries to the HAN. A query may be issued non-specifically for any IEEE 2030.5 device, or an IEEE 2030.5 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 8.5 (end device) and 8.6 (function set assignments).

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

8.3.4 LogEvents

There are no LogEvents generated by this function set.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

8.4 Self Device function set

8.4.1 Overview

The Self Device function set provides an interface for servers to publish general information about themselves (e.g., SFDI, LFDI, software version numbers).

8.4.2 List ordering

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

8.4.3 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 IEEE 2030.5 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.

8.4.4 LogEvents

Table 20—Self Device LogEvents

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

8.5 End Device function set

8.5.1 Overview

The End Device 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 End Device 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 End Device List.

Copyright © 2018 IEEE. All rights reserved.

8.5.2 List ordering

Table 21—EndDevice list ordering

Resource name	Primary key	Secondary key	Tertiary key
EndDevice	changedTime (descending)	SFDI (ascending)	href (ascending)

8.5.3 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, PowerStatus) 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.

Servers MAY remove records after 72 hours of no interaction from the corresponding client. It is RECOMMENDED that clients interact with the server 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. Servers SHALL NOT remove EndDevice instances before marking them as disabled (using the enabled attribute) and allowing time for clients to query this change. 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 IEEE 2030.5 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

8.5.4 LogEvents

Table 22—EndDevice LogEvents

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

8.6 Function Set Assignments function set

8.6.1 Overview

The Function Set Assignments function set defines collections of references to function set instances. These collections are used by the EndDevice 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).

8.6.2 List ordering

Table 23—Function Set Assignments list ordering

Resource name	Primary key	Secondary key	Tertiary key
FunctionSetAssignments	mRID (descending)	N/A	N/A

8.6.3 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, home energy management systems (HEMSs), 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 standard. 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 one Function Set Assignments for each HAN device registered to the server. A server SHOULD support three Function Set Assignments for each HAN device. Clients SHALL support at least six function set instances (e.g., two DemandResponseProgram instances, one Time instance, one TariffProfile instance, and two MessagingProgram instances) assigned through one 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Clients SHALL identify which Function Set Assignments apply to them by querying the FunctionSetAssignments resource within their End Device function set instance (see 8.5, end device function set, 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.

8.6.4 LogEvents

There are no LogEvents generated by this function set.

8.7 Subscription/Notification function set

8.7.1 Overview

This subclause describes the design and use of resources that support a generic, lightweight Subscription/Notification mechanism for use throughout the standard. 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.

8.7.2 List ordering

Notification resources typically are not exposed to other devices on the network, but devices that do choose to expose the resource(s) MUST obey the list ordering rules below.

Table 24—Subscription/Notification list ordering

Resource name	Primary key	Secondary key	Tertiary key
Subscription	href (ascending)	N/A	N/A
Notification	href (ascending)	N/A	N/A

8.7.3 Application guidelines/behavior

8.7.3.1 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.

8.7.3.2 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 standard 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.

8.7.3.3 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.

8.7.3.4 Subscription rules

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

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

b) Servers MAY remove Subscriptions at any time.

c) Servers SHOULD remove Subscriptions if a client has not renewed in 36 hours.

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

e) Servers SHALL use the Subscription parameters from the latest Subscription renewal.

f) Clients SHOULD check the status of their Subscriptions after perceived loss of connectivity.

g) Following recovery from a perceived loss of connectivity, clients SHOULD poll their resources of interest (including those to which they are subscribed) in case those resources changed during the loss of connectivity.

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

i) For Subscriptions to non-list resources, a Notification SHALL be sent whenever the representation of the non-list resource changes. If a resource contains a link, changes to the attributes of the link

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

(e.g., the “all” attribute of a ListLink) are not considered changes to the representation of the resource and therefore do not trigger a Notification.

j) 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. If a subordinate resource contains a link, changes to the attributes of the link (e.g., the “all” attribute of a ListLink) are not considered changes to the representation of the subordinate resource and therefore do not trigger a Notification. 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 4.7 for more information.

   1) 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.

k) 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. Note that, as a consequence of this rule, Notifications will not be sent for each change of a resource or threshold crossing.

l) Servers implementing the Subscription resource SHALL be capable of maintaining a minimum of one Subscription and servers implementing the Subscription resource SHOULD support at least one Subscription per device per subscribable resource.

m) 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.

n) 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 o) 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.

o) 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.

p) Servers SHOULD allow only the end device that corresponds to a given EndDevice resource to modify the Subscriptions within that resource.

q) The default recommended policy is that Subscription management SHOULD be performed using TLS.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

8.7.4 LogEvents

Table 25—Subscription/Notification LogEvents

LogEvent name	LogEvent code	LogEvent description
SUB_NTFY_FAIL	0x00	SHOULD be issued when there is a failure to successfully send a Notification (after a successful connection)
SUB_CONN_ESTB_FAIL	0x01	SHOULD be issued when there is a failure to successfully establish a connection to send a Notification

8.8 Response function set

8.8.1 Overview

This function set provides an interface for capturing responses from all events, including demand response and load control (DRLC), distributed energy resources (DERs), pricing, and messaging. Client devices of this function set include smart thermostats, in-premises displays, energy management systems, and devices that support load control, pricing, or text events.

The originating event will indicate if a response is required.

8.8.2 List ordering

Table 26—Response list ordering

Resource name	Primary key	Secondary key	Tertiary key
ResponseSet	mRID (descending)	N/A	N/A
Response	createdDateTime (descending)	endDeviceLFDI (ascending)	N/A

8.8.3 Application guidelines/behavior

8.8.3.1 Introduction

Clients of any function set that can utilize the Response function set (e.g., DRLC, DER, Price, Messaging, Flow Reservation) SHALL implement a Response function set client.

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.

Copyright © 2018 IEEE. All rights reserved.

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. Table 27 indicates the valid Response.status values supported by each function set.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyright © 2018 IEEE. All rights reserved.

Table 27—Response types by function set

Enumeration value	Description	Why sent	When sent	*response Required* bit position	DRLC and DER	Pricing	Messaging
0	*Reserved*						
1	Event received	To notify response server that client has initially received event	When the device first receives the event, either via a GET or a Notification	0	X	X	X
2	Event started	To notify response server that client has begun event	At *EffectiveStartTime* (see 10.2.3.2)	1	X	X	X
3	Event completed	To notify response server that the event fully and successfully completed	At *EffectiveEndTime* (see 10.2.3.2)	1	X	X	X
4	User has chosen to opt-out	To notify response server that user has chosen to opt out of the event. Can occur before event begins	At time user actively chose to opt out or when device automatically opts out due to user preference	1	X		
5	User has chosen to opt-in	To notify response server that user has chosen to opt in to the event (after a previous opt-out). Can occur before event begins	At time user actively chose to opt in or when device automatically opts in due to user preference	1	X		
6	The event has been cancelled	To notify response server that client has initially received cancellation	When the device first receives the cancellation, either via a GET or a Notification, even if the event has not yet begun execution	1	X	X	X
7	The event has been superseded	To notify response server that client has learned of an event that supersedes this event	When the device first learns of an event that supersedes this event, even if the event has not yet begun execution	1	X	X	X
8	Event partially completed with user opt-out	To notify response server that some participation in the event occurred, but not complete participation, due to one or more user opt-outs	At *EffectiveEndTime* (see 10.2.3.2)	1	X		
9	Event partially completed due to user opt-in	To notify response server that some participation in the event occurred, but not complete participation, due to one or more user opt-ins (after a previous opt-out)	At *EffectiveEndTime* (see 10.2.3.2)	1	X		

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 27—Response types by function set *(continued)*

Enumeration value	Description	Why sent	When sent	*response Required* bit position	DRLC and DER	Pricing	Messaging
10	Event completed, no user participation (previous opt-out)	To notify response server that no participation in the event occurred	At *EffectiveEndTime* (see 10.2.3.2)	1	X		
11	User has acknowledged the event	To notify response server that the client displayed the event and a human user actively acknowledged it	At time user actively chose to acknowledge the event	2	X	X	X
12	Cannot be displayed	To notify response server that the client is unable to display the message	When the device first receives the event, either via a GET or a Notification	1			X
13	Event aborted due to alternate server event	To notify response server that the client has chosen to stop execution of the event and instead execute an event from a different function set instance on a different server	At time the original event was ceased	1	X	X	X
14	Event aborted due to alternate program event	To notify response server that the client has chosen to stop execution of the event and instead execute an event from a different function set instance on the same server	At time the original event was ceased	1	X	X	X
15 to 251	*Reserved*						
252	Rejected: control parameter not applicable	To notify response server that client is unable to execute the event due to the controls being inapplicable to the device (e.g., a temperature offset was sent to a pool pump)	When the device first receives the event, either via a GET or a Notification	1	X		
253	Rejected: invalid event	To notify response server that client is unable to execute the event due to the controls (e.g., duty cycle, offset, setpoint) being out of range or not possible for the device (e.g., settings already at a maximum or minimum)	When the device first receives the event, either via a GET or a Notification	1	X		
254	Rejected: event was received after it expired	To notify response server that client has initially received event, but that the event was received after *SpecifiedEndTime*	When the device first receives the event, either via a GET or a Notification	1	X	X	X
255	*Reserved*						

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

8.8.3.2 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, the server SHALL minimally support one response for each function set for which it accepts responses.

8.8.4 LogEvents

Table 28—Response LogEvents

LogEvent name	LogEvent code	LogEvent description
RESPONSE_LIST_OVERFLOW	0x00	Should be issued by a server anytime their response list has exceeded maximum storage
RESPONSES_REPORTED_UPSTREAM	0x01	Should be issued by a server when responses are sent upstream

9. Common resources

9.1 Introduction

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

9.2 Time function set

9.2.1 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.).

9.2.2 List ordering

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.2.3 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 standard, 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 the supplemental material of IEEE Std 2030.5 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:

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

b) 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 backoff prior to establishing a valid time source. These requirements apply on a per time server basis.

Copyright © 2018 IEEE. All rights reserved.

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).

9.2.4 LogEvents

Table 29—Time LogEvents

LogEvent name	LogEvent code	LogEvent description
TM_TIME_ADJUSTED	0x00	Should be issued by a time server when its time is adjusted
TM_TIME_SOURCE_CHANGED	0x01	Should be issued by a time client when it chooses a different primary time source

9.3 Device Information function set

9.3.1 Overview

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

9.3.2 List ordering

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

9.3.3 Application guidelines/behavior

The Device Information function set is used to provide information about the actual device. Any device that implements an 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 an HTTP server are not required to serve the Device Information function set. However, this information MAY be posted to the appropriate EndDevice resource on an End Device function set server.

9.3.4 LogEvents

There are no LogEvents generated by this function set.

9.4 Power Status function set

9.4.1 Overview

The Power Status function set provides information regarding a device’s current power source, as well as basic status regarding any battery installed within the device.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.4.2 List ordering

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

9.4.3 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.

9.4.4 LogEvents

Table 30—Power Status LogEvents

LogEvent name	LogEvent code	LogEvent description
PS_LOW_BATTERY	0x00	SHOULD be issued when the installed battery charge drops below the configured low battery charge threshold
PS_LOW_BATTERY_RTN	0x01	SHOULD be issued when the installed battery charge rises above the configured low battery charge threshold (return to normal)

9.5 Network Status function set

9.5.1 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: CableLabs [B1], IETF RFC 2863, IETF RFC 3635 [B6], IETF RFC 4293 [B7], and Linux [B9].

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).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.5.2 List ordering

Table 31—Network Status list ordering

Resource name	Primary key	Secondary key	Tertiary key
IPInterface	ifIndex (ascending)	N/A	N/A
IPAddr	address (ascending)	N/A	N/A
LLInterface	EUI64 (ascending)	N/A	N/A
RPLInstance	DODAGid (ascending)	RPLInstanceID (ascending)	N/A
RPLSourceRoutes	SourceRoute (ascending)	DestAddress (ascending)	N/A

9.5.3 Application guidelines/behavior

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

9.5.4 LogEvents

Table 32—Network Status LogEvents

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

9.6 Log Event function set

9.6.1 Overview

The Log Event function set 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.6.2 List ordering

Table 33—LogEvent list ordering

Resource name	Primary key	Secondary key	Tertiary key
LogEvent	createdDateTime (descending)	LogEventID (descending)	N/A

9.6.3 Application guidelines/behavior

9.6.3.1 Introduction

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 defined by this standard or vendor defined.

9.6.3.2 LogEvents

There are no LogEvents generated by this function set.

The table below presents the LogEvent codes allocated by this standard for general IEEE 2030.5 events. These codes are associated with the value of zero (General) in the functionSet field.

Table 34—General LogEvents

LogEvent name	LogEvent code	LogEvent description
LE_GEN_HARDWARE	0x00	SHOULD be issued when an unclassified general hardware fault occurs
LE_GEN_HARDWARE_RTN	0x01	SHOULD be issued when an unclassified general hardware fault is corrected (return to normal)
LE_GEN_SOFTWARE	0x02	SHOULD be issued when an unclassified general software fault occurs
LE_GEN_SOFTWARE_RTN	0x03	SHOULD be issued when an unclassified general software fault is corrected (return to normal)

NOTE—See the function set definitions for LogEvent codes associated with each function set.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.7 Configuration function set

9.7.1 Overview

The Configuration function set 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 IETF RFC 3535 [B5]. 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 an IEEE 2030.5 implementation.

9.7.2 List ordering

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

9.7.3 Application guidelines/behavior

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

9.7.4 LogEvents

Table 35—Configuration LogEvents

LogEvent name	LogEvent code	LogEvent description
CFG_CONFIGURATION_UPDATED	0x00	SHALL be issued when the local Configuration resource is changed

9.8 File Download function set

9.8.1 Introduction

This subclause describes the mechanisms used to support secure, interoperable, remote software download to IEEE 2030.5 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, IEEE 2030.5 supports only a polled mode of file download.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.8.2 File list resource

9.8.2.1 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.

9.8.2.2 List ordering

File resources are ordered within a FileList as follows:

Table 36—File download list ordering

Resource name	Primary key	Secondary key	Tertiary key	Quaternary key
File	mfID (ascending)	mfModel (ascending)	mfVer (descending)	href (ascending)

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.

9.8.2.3 Application guidelines/behavior

9.8.2.3.1 Introduction

This standard 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 time. 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 standard.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

IEEE 2030.5 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 IETF RFC 7232, 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 five 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 five 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.

9.8.2.3.2 File formats

IEEE Std 2030.5 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 standard.

IEEE 2030.5 does, however, require a file be digitally signed to protect the file from undetected modification and provide file origin authentication.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.8.2.3.3 File signatures

Files SHALL be signed and signatures verified using approved algorithms specified in NIST SP800-131A. Cryptographic strength MUST be at least a minimum of 128-bit. This standard 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. The signature of the file SHALL be calculated over the entire contents of the file excluding the signature octets.

9.8.2.3.4 Loading files containing security credentials

File loads containing IEEE 2030.5 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.

9.8.2.3.5 File query parameters

Beyond the standard IEEE 2030.5 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:

- type
- lFDI
- mfHwVer
- mfID
- mfModel
- mfSerNum
- 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 the supplemental material of IEEE Std 2030.5.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.8.2.4 LogEvents

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

Table 37—FileDownload LogEvents

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

9.8.3 File status resource

9.8.3.1 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.

9.8.3.2 List ordering

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

9.8.3.3 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 the supplemental material of IEEE Std 2030.5).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9.8.3.4 LogEvents

There are no LogEvents generated by this resource.

10. Smart energy resources

10.1 Introduction

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

10.2 Common application functionality

10.2.1 Overview

This subclause 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.

10.2.2 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.

10.2.3 Event rules and guidelines

10.2.3.1 Introduction

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.

10.2.3.2 Definitions

The following definitions define terms that will be used throughout the event rules and guidelines subclause. 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 (IEEE Std 2030.5 supplemental material).

Copyright © 2018 IEEE. All rights reserved.

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 any applied Duration Randomization offsets (which may be a positive or negative value) 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* from the same function set, regardless of the controls present in the *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*.

Override: A local act by the end user (e.g., a button press).

10.2.3.3 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 DRLC and DER. Pricing is somewhat unique in that it exerts control-like effects upon price-responsive clients; however, it is not a

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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:

a) 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.

b) 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 a) above; additional polling might not be needed.

c) 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*.

d) For function sets with direct control, Flow Reservation Responses with the same subject mRID, 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.

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

f) 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) creationTime is the newer *Event*.

g) *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*.

h) When changing conditions mandate changes in the sequence or contents of *Events*, the following guidelines MAY be used to indicate desired actions:

    1) Canceling existing *Events* and reissuing new *Events*.

    2) Sending overlapping or nested *Events* to supersede existing *Events*.

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

j) For function sets with direct control, it is RECOMMENDED that process h)2) 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 g).

k) 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*.

l) 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.

    1) 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".

m) 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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*.

n) 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:

   1) 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.

   2) 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*.

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

   1) 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.

   2) Randomization SHALL NOT artificially create a gap between *Successive Events*.

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

q) If multiple deviceCategory's are identified for an *Event*, future *Events* for an individual deviceCategory (or a subset of the original *Event*) that cause an *Overlapping Event* will supersede the original *Event* strictly for that deviceCategory (or a subset of the original *Event*). Note: Rule f) applies to all *Overlapping Events*.

   1) Those clients whose deviceCategory is not listed in the future *Event* but whose deviceCategory was included in the original *Event* SHALL continue to execute per the parameters of the original *Event*.

   2) Rule c) continues to apply. Servers SHALL NOT edit the original *Event* but SHALL maintain all *Events* in their entirety.

   3) A server SHALL set the potentiallySuperseded flag when the *Event* is superseded for any of the device categories and update the potentiallySupersededTime.

r) 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.

s) 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.

t) For DERControls, differing controls (e.g., opModTargetVar, opModTargetW) within DERControl *Events* are independent and are allowed to overlap or nest without superseding. If multiple controls are identified for a DERControl *Event*, future DERControl *Events* for an individual control (or a subset of the original *Event*) that cause an *Overlapping Event* will supersede the original *Event* strictly for that control (or a subset of the original *Event*). Note: Rule f) applies to all *Overlapping Events*.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

1) Those clients whose control is not listed in the future *Event* but whose control was included in the original *Event* SHALL continue to execute per the parameters of the original *Event*.
2) Those clients whose control is listed in the future *Event* but whose deviceCategory is not listed in the future *Event* SHALL continue to execute per the parameters of the original *Event*.
3) Rule c) continues to apply. Servers SHALL NOT edit the original *Event* but SHALL maintain all *Events* in their entirety.
4) A server SHALL set the potentiallySuperseded flag when the *Event* is superseded for any of the controls and update the potentiallySupersededTime.
5) This rule also applies to the DefaultDERControl and DERControl *Events* should always be assumed to overlap the DefaultDERControl.

10.2.4 Randomization

10.2.4.1 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 the distribution infrastructure for the related commodity (e.g., an electrical distribution network).

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. Note that other standards may be applicable to the underlying device and may provide their own protections that render randomization unnecessary (e.g., IEEE Std 1547™).

10.2.4.2 Randomization attributes

10.2.4.2.1 Introduction

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 C.21.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.2.4.2.2 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 scheduled *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 11:00 AM and has a *randomizeStart* value of "-300" (i.e., five minutes prior randomization), a client could randomly generate a number of "-120" (i.e., two minutes prior randomization) and begin the *Event* at 10:58 AM. 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*.

10.2.4.2.3 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 1:00 PM and has a randomizeDuration value of "−180," a client can select to end the *Event* at 12:59 PM, 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*.

10.2.4.3 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 pseudorandom values SHALL scale the applied randomization based on the range indicated by the given randomizeStart or randomizeDuration. Stated differently, fixed pseudorandom values SHALL indicate a percentage of the randomizeStart or randomizeDuration to be applied. Fixed pseudorandomization is when a given device has a value that is applied for all randomizations. Manufacturers SHALL assure that fixed pseudorandomization values are random over a population of like devices.

Manufacturers SHALL assure that random number generators maintain a distribution of values over a population of like devices (e.g., by using differing seeds).

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 the absolute value of the randomizeStart attribute and the absolute value of the randomizeDuration to the abbreviated *Event*.

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.2.5 Multi-server

10.2.5.1 Overview

This subclause 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.

10.2.5.2 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 Clause 6 (security), 8.6 (function set assignments), and Clause 7 (service discovery) 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.

10.2.5.3 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 8.6 (time function set) for guidelines on dealing with multiple time servers in the network.

10.2.5.4 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 10.6 (messaging function set).

10.2.5.5 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 10.5 (Pricing function set).

10.2.5.6 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.

Copyright © 2018 IEEE. All rights reserved.

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:

a) Servers SHALL indicate their primacy in the primacy element of the function set instance. See schema (IEEE Std 2030.5 supplemental material) definition of PrimacyType for possible values.

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

   1) 0 supersedes 1

   2) 1 supersedes 2

   3) 2 supersedes 3

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

c) 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 Server Event” Response.status for the *Event* that was superseded. Likewise, if a client is executing an *Event* from one program and decides to execute an *Event* from a different program on the same server per the application guidelines and if the superseded *Event* responseRequired indicates, the client SHALL send a response with “Aborted due to an Alternate Program Event” Response.status for the *Event* that was superseded.

This mechanism and these guidelines allow service providers to coordinate among 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.

10.3 Demand Response and Load Control function set

10.3.1 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). EDC instances can expose attributes that allow devices to respond to events that are explicitly targeted at

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

specific kinds of devices. 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.

10.3.2 List ordering

Table 38—Demand Response and Load Control list ordering

Resource name	Primary key	Secondary key	Tertiary key
DemandResponseProgram	Primacy (ascending)	mRID (descending)	N/A
EndDeviceControl and ActiveEndDeviceControl	interval.start (ascending)	creationTime (descending)	mRID (descending)
LoadShedAvailability	href (ascending)	N/A	N/A

10.3.3 Application guidelines/behavior

10.3.3.1 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 client 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.

10.3.3.2 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 of 10 unique EndDeviceControl instances.

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.3.3.3 Rules and guidelines

10.3.3.3.1 Introduction

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.

10.3.3.3.2 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 provide their LoadShedAvailability for that program. If the server provides both, the client SHALL update its current load shed availability when either threshold is crossed.

10.3.3.3.3 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.”

10.3.3.3.4 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.

10.3.3.3.5 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).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.3.3.3.6 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."

10.3.3.4 Response

The properties of the various control values in a DrResponse with a status of "Event received" SHALL be those existing at the time the event was received but not yet started.

The properties of the various control values in a DrResponse with a status of "Event started" SHALL be those existing at the time the event has started (i.e., the values to which the client adjusted).

The properties of the various control values in a DrResponse with a status indicating event termination (e.g., "Event completed," "User has chosen to opt-out," "The event has been superseded") SHALL be those existing after the event's completion. For example, if an event completes and no other event has started, the properties of the various control values will likely be identical to those that existed prior to the event.

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 energy management systems (EMSs) 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 8.8.

10.3.3.5 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 under the appropriate EndDevice resource or SHALL provide their ability to shed load locally under their SelfDevice 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 provide updates to their LoadShedAvailability for the provider of that program.

Clients reporting their LoadShedAvailability to multiple DemandResposePrograms 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 2 kW load SHALL NOT report 2 kW 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.

10.3.4 LogEvents

There are no LogEvents generated by this function set.

Copyright © 2018 IEEE. All rights reserved.

10.4 Metering function set

10.4.1 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 Annex 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.

10.4.2 List ordering

Table 39—Metering list ordering

Resource name	Primary key	Secondary key	Tertiary key
UsagePoint	mRID (descending)	N/A	N/A
MeterReading	mRID (descending)	N/A	N/A
ReadingSet	timePeriod.start (descending)	mRID (descending)	N/A
Reading	localID (ascending)	consumptionBlock (ascending)	touTier (ascending)

10.4.3 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. Only relevant IEC fields are listed in the XML Schema described in the supplemental material of IEEE Std 2030.5 and IEEE Std 2030.5 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. 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 0xFFFFFFFFFFFFFFFFFFFFFFFF[XXXXXXXX] (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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 40.
- 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 40.
  - 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 ReadingSet 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. Note that if the number of TOU tiers is 1, using a value of 1 instead of 0 will be less efficient and is DISCOURAGED.
    - 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 Reading Type resource matching the appropriate definition in Table 40, and with the Reading values corresponding to the appropriate data type.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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

a) SHALL contain a ReadingTypeLink element that points to a ReadingType resource that matches an Interval data definition from Table 40.

    1) The ReadingType resources SHALL specify the intervalLength that is the default for the intervals contained in the ReadingList resource.

b) SHALL contain a ReadingSetListLink element that points to a ReadingSetList resource.

    1) When metering data is present, the ReadingSetList SHALL contain at least one ReadingSet source, which MAY correspond to the present Interval Block data (the one currently being filled).

        i) 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.

        ii) If the duration in the timePeriod of the Reading is not equal to the intervalLength specified in the Reading Type the timePeriod SHALL be included in the Reading.

c) SHALL NOT contain a ReadingLink resource.

Table 40 provides a list of common Reading Type Definitions with related fields listed.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 40—Reading types

ReadingType Element	accumulationBehavior	calorificValue	commodity	conversionFactor	dataQualifier	flowDirection	intervalLength	kind	numberOfConsumptionBlocks	numberOfTouTiers	phase	powerOfTenMultiplier	subIntervalLength	suppyLimit	tieredConsumptionBlocks	uom
Instantaneous demand	12		1			1	E	8				O	E		E	38
Summation delivered	9		1			1	E	12	O	O		O	E			72
Summation received	9		1			19	E	12	O	O		O	E			72
Maximum demand delivered	6		1		8	1	O	8	O	O		O		E		38
Maximum demand received	6		1		8	19	O	8	O	O		O		E		38
Intervals delivered	4		1			1	O	12				O	E		E	72
Intervals received	4		1			19	O	12				O	E		E	72

Blank cells indicate not used for the given ReadingType name. May be used for other ReadingTypes.
“E” indicates they are not to be used for this class of ReadingType no matter commodity.
“O” indicates that there could 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 (U.S. gallon) or 134 (liters). Other commodities are to 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 the supplemental material of IEEE Std 2030.5.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.4.4 LogEvents

This subclause includes definitions of all LogEvents that may be raised by the Metering function set. The LogEvent names and codes are summarized in Table 41.

Table 41 —Metering LogEvents

LogEvent name	LogEvent code	LogEvent description
UPT_CHECK_METER	0x00	SHOULD be issued when check meter alarm occurs
UPT_CHECK_METER_RTN	0x01	SHOULD be issued when check meter alarm clears
UPT_TAMPER_DETECT	0x02	SHOULD be issued when a tampering is detected
UPT_TAMPER_DETECT_RTN	0x03	SHOULD be issued when a tampering detect clears
UPT_POWER_QUALITY	0x04	SHOULD be issued when power quality alarm occurs. It is a generic power quality event code.
UPT_POWER_QUALITY_RTN	0x05	SHOULD be issued when power quality alarm clears
UPT_ LEAK_DETECT	0x06	SHOULD be issued when a leak is detected
UPT_LEAK_DETECT_RTN	0x07	SHOULD be issued when a leak detect clears
UPT_SERVICE_DISCONNECT	0x08	SHOULD be issued when service is disconnected
UPT_SERVICE_CONNECT	0x09	SHOULD be issued when service is connected
UPT_SERVICE_LIMITED	0x0A	SHOULD be issued when service limited alarm occurs
UPT_SERVICE_LIMITED_RTN	0x0B	SHOULD be issued when service limited alarm clears
UPT_LOW_VOLTAGE_L1	0x0C	SHOULD be issued when low voltage L1 occurs
UPT_LOW_VOLTAGE_L1_RTN	0x0D	SHOULD be issued when low voltage L1 clears
UPT_HIGH_VOLTAGE_L1	0x0E	SHOULD be issued when high voltage L1 occurs
UPT_HIGH_VOLTAGE_L1_RTN	0x0F	SHOULD be issued when high voltage L1 clears
UPT_LOW_VOLTAGE_L2	0x10	SHOULD be issued when low voltage L2 occurs
UPT_LOW_VOLTAGE_L2_RTN	0x11	SHOULD be issued when low voltage L2 clears
UPT_HIGH_VOLTAGE_L2	0x12	SHOULD be issued when high voltage L2 occurs
UPT_HIGH_VOLTAGE_L2_RTN	0x13	SHOULD be issued when high voltage L2 clears
UPT_LOW_VOLTAGE_L3	0x14	SHOULD be issued when low voltage L3 occurs
UPT_LOW_VOLTAGE_L3_RTN	0x15	SHOULD be issued when low voltage L3 clears
UPT_HIGH_VOLTAGE_L3	0x16	SHOULD be issued when high voltage L3 occurs
UPT_HIGH_VOLTAGE_L3_RTN	0x17	SHOULD be issued when high voltage L3 clears
UPT_OVER_CURRENT_FORWARD_L1	0x18	SHOULD be issued when overcurrent L1 occurs in the forward (consumed by premises) direction
UPT_OVER_CURRENT_FORWARD_L1_RTN	0x19	SHOULD be issued when overcurrent L1 clears in the forward (consumed by premises) direction
UPT_OVER_CURRENT_FORWARD_L2	0x1A	SHOULD be issued when overcurrent L2 occurs in the forward (consumed by premises) direction
UPT_OVER_CURRENT_FORWARD_L2_RTN	0x1B	SHOULD be issued when overcurrent L2 clears in the forward (consumed by premises) direction
UPT_OVER_CURRENT_FORWARD_L3	0x1C	SHOULD be issued when overcurrent L3 occurs in the forward (consumed by premises) direction
UPT_OVER_CURRENT_FORWARD_L3_RTN	0x1D	SHOULD be issued when overcurrent L3 clears in the forward (consumed by premises) direction
UPT_FREQUENCY_TOO_LOW_L1	0x1E	SHOULD be issued when frequency too low L1
UPT_FREQUENCY_TOO_LOW_L1_RTN	0x1F	SHOULD be issued when frequency too low L1 clears
UPT_FREQUENCY_TOO_HIGH_L1	0x20	SHOULD be issued when frequency too high L1
UPT_FREQUENCY_TOO_HIGH_L1_RTN	0x21	SHOULD be issued when frequency too high L1 clears
UPT_FREQUENCY_TOO_LOW_L2	0x22	SHOULD be issued when frequency too low L2
UPT_FREQUENCY_TOO_LOW_L2_RTN	0x23	SHOULD be issued when frequency too low L2 clears
UPT_FREQUENCY_TOO_HIGH_L2	0x24	SHOULD be issued when frequency too high L2
UPT_FREQUENCY_TOO_HIGH_L2_RTN	0x25	SHOULD be issued when frequency too high L2 clears

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 41—Metering LogEvents *(continued)*

LogEvent name	LogEvent code	LogEvent description
UPT_FREQUENCY_TOO_LOW_L3	0x26	SHOULD be issued when frequency too low L3
UPT_FREQUENCY_TOO_LOW_L3_RTN	0x27	SHOULD be issued when frequency too low L3 clears
UPT_FREQUENCY_TOO_HIGH_L3	0x28	SHOULD be issued when frequency too high L3
UPT_FREQUENCY_TOO_HIGH_L3_RTN	0x29	SHOULD be issued when frequency too high L3 clears
UPT_GROUND_FAULT	0x2A	SHOULD be issued when ground fault occurs
UPT_GROUND_FAULT_RTN	0x2B	SHOULD be issued when ground fault clears
UPT_BURST_DETECT	0x2C	SHOULD be issued when burst detect alarm occurs
UPT_BURST_DETECT_RTN	0x2D	SHOULD be issued when burst detect alarm clears
UPT_PRESSURE_TOO_LOW	0x2E	SHOULD be issued when pressure too low
UPT_PRESSURE_TOO_LOW_RTN	0x2F	SHOULD be issued when pressure too low clears
UPT_PRESSURE_TOO_HIGH	0x30	SHOULD be issued when pressure too high
UPT_PRESSURE_TOO_HIGH_RTN	0x31	SHOULD be issued when pressure too high clears
UPT_FLOW_SENSOR_COMMUNICATION_ ERROR	0x32	SHOULD be issued when flow sensor communication error occurs
UPT_FLOW_SENSOR_COMMUNICATION_ ERROR_RTN	0x33	SHOULD be issued when flow sensor communication error clears
UPT_FLOW_SENSOR_MEASUREMENT_ FAULT	0x34	SHOULD be issued when flow sensor measurement fault occurs
UPT_FLOW_SENSOR_MEASUREMENT_ FAULT_RTN	0x35	SHOULD be issued when flow sensor measurement fault clears
UPT_FLOW_SENSOR_REVERSE_FLOW	0x36	SHOULD be issued when reverse flow is detected
UPT_FLOW_SENSOR_REVERSE_FLOW _RTN	0x37	SHOULD be issued when reverse flow clears
UPT_FLOW_SENSOR_AIR_DETECT	0x38	SHOULD be issued when flow sensor air detect alarm occurs
UPT_FLOW_SENSOR_AIR_DETECT_RTN	0x39	SHOULD be issued when flow sensor air detect alarm clears
UPT_PIPE_EMPTY	0x3A	SHOULD be issued when pipe empty alarm occurs
UPT_PIPE_EMPTY_RTN	0x3B	SHOULD be issued when pipe empty alarm clears
UPT_INLET_TEMPERATURE_SENSOR_ FAULT	0x3C	SHOULD be issued when inlet temperature sensor fault
UPT_INLET_TEMPERATURE_SENSOR_ FAULT_RTN	0x3D	SHOULD be issued when inlet temperature sensor fault clears
UPT_OUTLET_TEMPERATURE_SENSOR_ FAULT	0x3E	SHOULD be issued when outlet temperature sensor fault
UPT_OUTLET_TEMPERATURE_SENSOR_ FAULT_RTN	0x3F	SHOULD be issued when outlet temperature sensor fault clears
UPT_OVER_CURRENT_REVERSE_L1	0x40	SHOULD be issued when overcurrent L1 occurs in the reverse (supplied by premises) direction
UPT_OVER_CURRENT_REVERSE_L1_RTN	0x41	SHOULD be issued when overcurrent L1 clears in the reverse (supplied by premises) direction
UPT_OVER_CURRENT_REVERSE_L2	0x42	SHOULD be issued when overcurrent L2 occurs in the reverse (supplied by premises) direction
UPT_OVER_CURRENT_REVERSE_L2_RTN	0x43	SHOULD be issued when overcurrent L2 clears in the reverse (supplied by premises) direction
UPT_OVER_CURRENT_REVERSE_L3	0x44	SHOULD be issued when overcurrent L3 occurs in the reverse (supplied by premises) direction
UPT_OVER_CURRENT_REVERSE_L3_RTN	0x45	SHOULD be issued when overcurrent L3 clears in the reverse (supplied by premises) direction

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.5 Pricing function set

10.5.1 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 the supplemental material of IEEE Std 2030.5, critical peak pricing (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.

10.5.2 List ordering

Table 42—Pricing list ordering

Resource name	Primary key	Secondary key	Tertiary key
TariffProfile	mRID (descending)	N/A	N/A
RateComponent	mRID (descending)	N/A	N/A
TimeTariffInterval and ActiveTimeTariffInterval	interval.start (ascending)	creationTime (descending)	mRID (descending)
ConsumptionTariffInterval	startValue (ascending)	N/A	N/A

10.5.3 Application guidelines/behavior

10.5.3.1 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).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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

10.5.3.2 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]).

10.5.3.3 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

Copyright © 2018 IEEE. All rights reserved.

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 TimeTariffInterval 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.

10.5.3.4 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.

10.5.3.5 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 instance MAY NOT include a ConsumptionTariffIntervalListLink, meaning that ConsumptionTariffIntervals (and therefore the actual price) are 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.5.3.6 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 minutes to 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.

10.5.3.7 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 systems) MAY follow multiple Pricing servers and make policy-based decisions to dispatch local resources (e.g., distributed energy resources) 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.

10.5.3.8 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 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.5.3.9 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 10.5.3.8).

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.

10.5.4 LogEvents

Table 43—Pricing LogEvents

LogEvent name	LogEvent code	LogEvent description
TP_NO_TTI	0x00	SHOULD be generated by a Pricing client when there is a gap between two TimeTariffInterval instances or if a TimeTariffInterval instance completes and is not followed by another.

10.6 Messaging function set

10.6.1 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.

Copyright © 2018 IEEE. All rights reserved.

10.6.2 List ordering

For list ordering purposes, the MessagingProgramList, TextMessageList, and the ActiveTextMessageList SHALL be ordered based on the criteria in Table 44.

Table 44—Message list ordering

Resource name	Primary key	Secondary key	Tertiary key
MessagingProgram	mRID (descending)	N/A	N/A
TextMessage and ActiveTextMessage	dateTimeInterval.start (ascending)	createdDateTime (descending)	mRID (descending)

10.6.3 Application guidelines/behavior

10.6.3.1 MessagingProgram

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

10.6.3.2 TextMessage

MessagingProgram server devices SHALL be capable of internally storing and supporting at least one MessagingProgram instance. MessagingProgram server devices SHOULD be capable of internally storing and supporting three unique TextMessage instances. Additional information for common application features that the Messaging function set will perform can be found in 10.2, the common application functionality section of this document.

Messaging client devices SHALL be capable of internally storing and supporting at least one 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 function 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 message’s priority.

10.6.4 LogEvents

There are no LogEvents generated by this function set.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.7 Billing function set

10.7.1 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 backend 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.7.2 List ordering

Table 45—Billing list ordering

Resource name	Primary key	Secondary key	Tertiary key
CustomerAccount	customerName (ascending)	mRID (descending)	N/A
CustomerAgreement	serviceLocation (ascending)	mRID (descending)	N/A
BillingPeriod	interval.start (descending)	href (ascending)	N/A
HistoricalReading	description (ascending)	mRID (descending)	N/A
TargetReading	description (ascending)	mRID (descending)	N/A
ProjectionReading	description (ascending)	mRID (descending)	N/A
BillingReadingSet	timePeriod.start (descending)	mRID (descending)	N/A
BillingReading	timePeriod.start (ascending)	consumptionBlock (ascending)	touTier (ascending)

10.7.3 Application guidelines/behavior

10.7.3.1 CustomerAccount and CustomerAgreement 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.

10.7.3.2 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.

10.7.3.3 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 100 kWh in a previous month with a reduction target of 10%. This target would be specified in the TargetReading as 90 kWh and it would be left to the device to calculate the percentage or kWh reduction to which this equates.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.7.3.4 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.

10.7.3.5 HistoricalReading resource

HistoricalReadings 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 back-office 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.). The targets in a percentage reduction or other method MUST be converted to an absolute value.

10.7.3.6 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.

10.7.4 LogEvents

There are no LogEvents generated by this function set.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.8 Prepayment function set

10.8.1 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 British thermal units [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 IEEE Std 2030.5 only provides a wrapper for this token data, the mechanism by which tokens are produced and consumed is out of scope for this standard).

10.8.2 List ordering

Table 46—Prepayment list ordering

Resource name	Primary key	Secondary key	Tertiary key
Prepayment	mRID (descending)	N/A	N/A
CreditRegister	effectiveTime (descending)	mRID (descending)	N/A
SupplyInterruptionOverride	interval.start (ascending)	interval.duration (ascending)	N/A

10.8.3 Application guidelines/behavior

10.8.3.1 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.,

Copyright © 2018 IEEE. All rights reserved.

emergency credit might only be allowed when availableCredit is less than or equal to the creditExpiryLevel).

10.8.3.2 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 in-premises display 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.

10.8.3.3 Mirroring behavior

In most IEEE 2030.5 function sets, mirroring involves behavior similar to the following pattern:

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

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

c) 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:

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

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

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

— 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.8.4 LogEvents

Table 47—Prepayment LogEvents

LogEvent name	LogEvent code	LogEvent description
PPY_CREDIT_POST_SUCCESS	0x00	SHOULD be issued when a client successfully posts a CreditRegister to the CreditRegisterList
PPY_CREDIT_POST_FAIL	0x01	SHOULD be issued when a client posts an invalid resource to the CreditRegisterList

10.9 Flow Reservation function set

10.9.1 Overview

This function set provides an interface for exchange of 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 ESI, electric vehicle supply equipment (EVSE), and EMS. 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.

10.9.2 List ordering

Table 48—FlowReservation list ordering

Resource name	Primary key	Secondary key	Tertiary key
FlowReservationRequest	interval.start (ascending)	creationTime (descending)	mRID (descending)
FlowReservationResponse	interval.start (ascending)	creationTime (descending)	mRID (descending)

10.9.3 Application guidelines/behavior

10.9.3.1 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. FlowReservation server

Copyright © 2018 IEEE. All rights reserved.

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.

Servers SHOULD remove FlowReservationRequests when the associated FlowReservationResponse expires. Servers SHALL retain FlowReservationRequests which have active FlowReservationResponses so that clients are able to cancel them if necessary.

10.9.3.2 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.

10.9.4 LogEvents

Table 49—Flow Reservation LogEvents

LogEvent name	LogEvent code	LogEvent description
FR_SCHEDULING_ERROR	0x00	SHOULD be issued if the server encounters an error and is unable to schedule reservations normally

10.10 Distributed Energy Resources function set

10.10.1 Overview

This function set provides an interface to manage distributed energy resources (DERs). 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 explicitly targeted at specific kinds of devices. 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 IEEE 2030.5 DER client model is based on the SunSpec Alliance Inverter Control Model (SunSpec [B10]) which is derived from IEC 61850-7-420 [B4] and EPRI [B2].

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.10.2 Terminology and conventions

The terminology used to describe the IEEE 2030.5 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 IEEE 2030.5 DER interface defines the power factor representation to avoid configuration mismatches. DER power factor settings representations consist of two elements: an unsigned value representing the ratio of active power to apparent power and an excitation. The excitation indicates whether reactive power is being injected (over-excited) or absorbed (under-excited).

It may be desirable to have different power factor settings when a DER is injecting active power and when it is absorbing active power. For this reason, the DER function set supports fixed power factor for both injecting active power and absorbing active power.

For measurement purposes, a power factor measurement is a single value that reflects the ratio of active power to apparent power. If a sign is present, it is the same as the active power sign. The sign of reactive power indicates the excitation (positive for over-excited and negative for under-excited).

These relationships are shown in Figure 3. 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

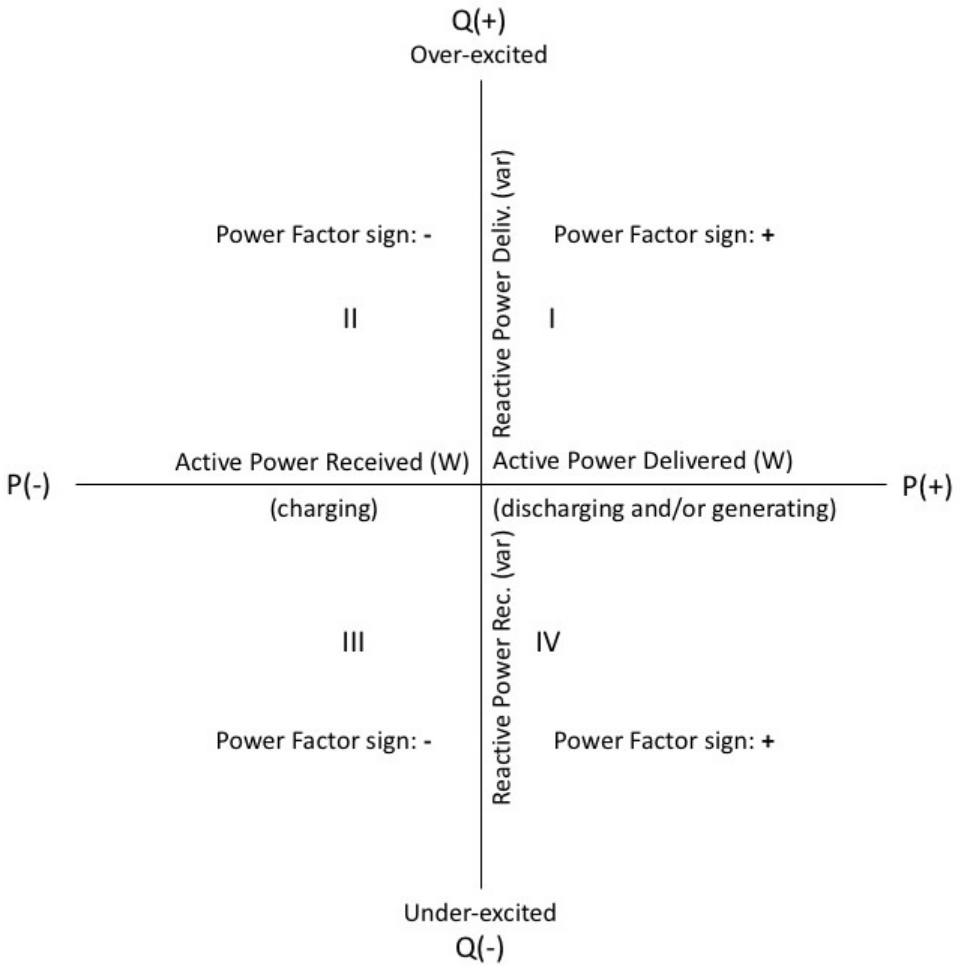


Figure 3—Active and reactive power flow directions as measured at the DER

10.10.3 List ordering

Table 50—Distributed energy resources list ordering

Resource name	Primary key	Secondary key	Tertiary key
DERProgram	primacy (ascending)	mRID (descending)	N/A
DERControl and ActiveDERControl	interval.start (ascending)	creationTime (descending)	mRID (descending)
DERCurve	creationTime (descending)	mRID (descending)	N/A
DER	href (ascending)	N/A	N/A

10.10.4 Application guidelines/behavior

10.10.4.1 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

10.10.4.2 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 applicable 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 applicable 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.

10.10.4.3 DERCurve

10.10.4.3.1 Introduction

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. Curve types define a single independent variable (xvalue) and dependent variable (yvalue) per curveData point. See the schema (IEEE Std 2030.5 supplemental material) for details on curve-based DER control modes.

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 4, 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

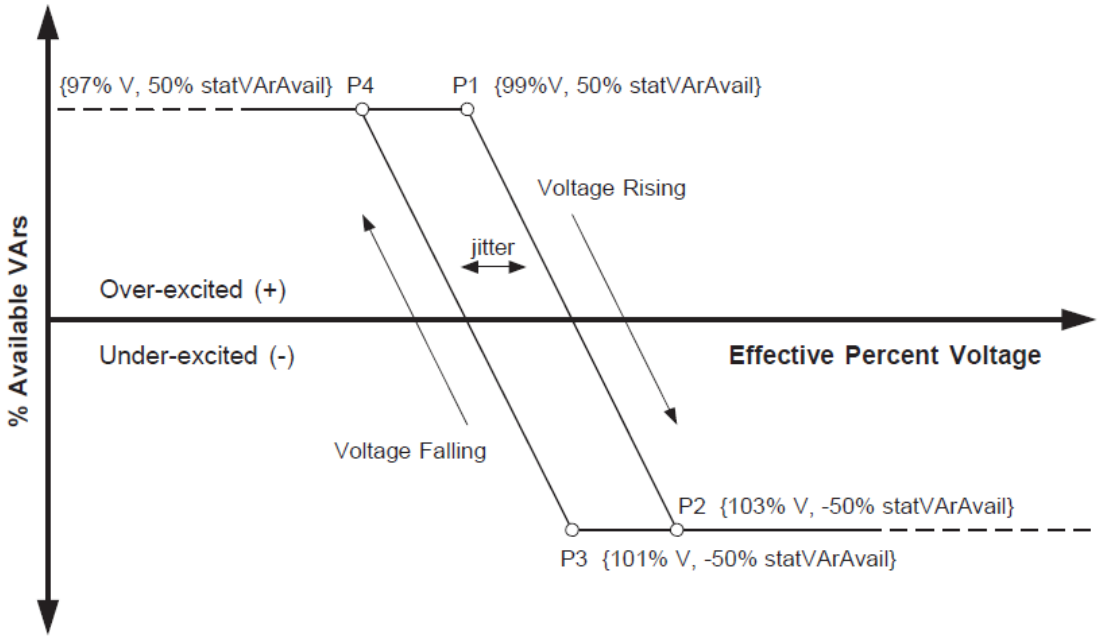


Figure 4—Example Volt-Var curve and hysteresis

10.10.4.3.2 Ride-through curve representation

Must trip, may trip, and momentary cessation curves are represented as piecewise linear curves that define the regions associated with voltage and frequency must trip, may trip, and momentary cessation behavior. It is desirable to use a mechanism to represent the curves that is flexible and handles as many use cases as possible.

The threshold requirements are represented by supplying a method to designate the following regions: must trip, may trip, and momentary cessation. Each region is defined with a piecewise linear curve demarcating the boundary (e.g., when crossing the may trip curve, the DER is in the may trip region).

The difference between must trip and momentary cessation is the process of resuming operation once that region has been entered. The exact resumption process may vary based on grid code and additional parameters, but the general distinction is that resumption from momentary cessation may be done fully and immediately on leaving the region while resumption from must trip may require additional considerations such as a delay and ramping operation. Due to the limits in some DER, galvanic isolation may or may not be provided on a must trip.

When the must trip region is entered, the DER SHALL trip.

When the may trip region is entered, the DER MAY continue in its current operational mode (either cease to energize or mandatory operation) or MAY trip.

When the momentary cessation region is entered, the DER SHALL cease to energize, but SHALL NOT trip.

DER curves have a defined hierarchy:

a) Must trip

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

b) Momentary cessation

c) May trip

When crossing a curve of higher precedence (must trip is the highest), the DER SHALL assume the behavior of the higher precedence.

A DER can be configured with only the must trip and momentary cessation curves (e.g., changing behavior one cycle prior to the must trip and momentary cessation curves) so long as the behavior is verified with the interconnection certification standard. The may trip curve can be useful if the DER makes use of the optional regions.

Curves are assumed to extend indefinitely vertically from the first point on the curve (positive voltage direction for high voltage and negative voltage direction for low voltage) and horizontally (positive time) from the last point on the curve.

It is recognized that most DER will have significant limitations on the shape of the curves that can be supported. Many DER may only be able to support curves with vertical and horizontal curve segments within very specific ranges.

Figure 5 shows an example of voltage must trip, momentary cessation, and may trip curves, and Table 51 shows the specific values that would comprise the curve settings.

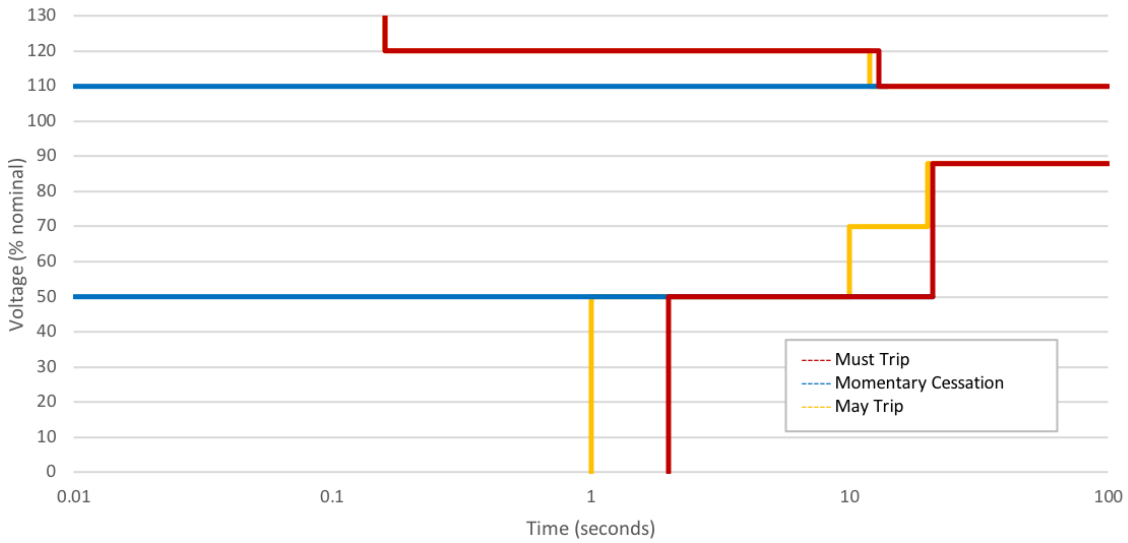


Figure 5—Example low and high-voltage ride-through curves

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 51—Example low and high-voltage ride-through curve values

Curve	Points
LV must trip	(2, 0), (2, 50), (21, 50), (21, 88), (100, 88)
LV momentary cessation	(0, 50), (1.5, 50)
LV may trip	(1, 0), (1, 50), (10, 50), (10, 70), (20, 70), (20, 88), (100, 88)
HV must trip	(0.16, 130), (0.16, 120), (13, 120), (13, 110), (100, 110)
HV momentary cessation	(0, 110), (13, 110)
HV may trip	(0.16, 130), (0.16, 120), (12, 120), (12, 110), (100, 110)

Figure 6 shows an example of frequency must trip and may trip curves, and Table 52 shows the specific values that would comprise the curve settings.

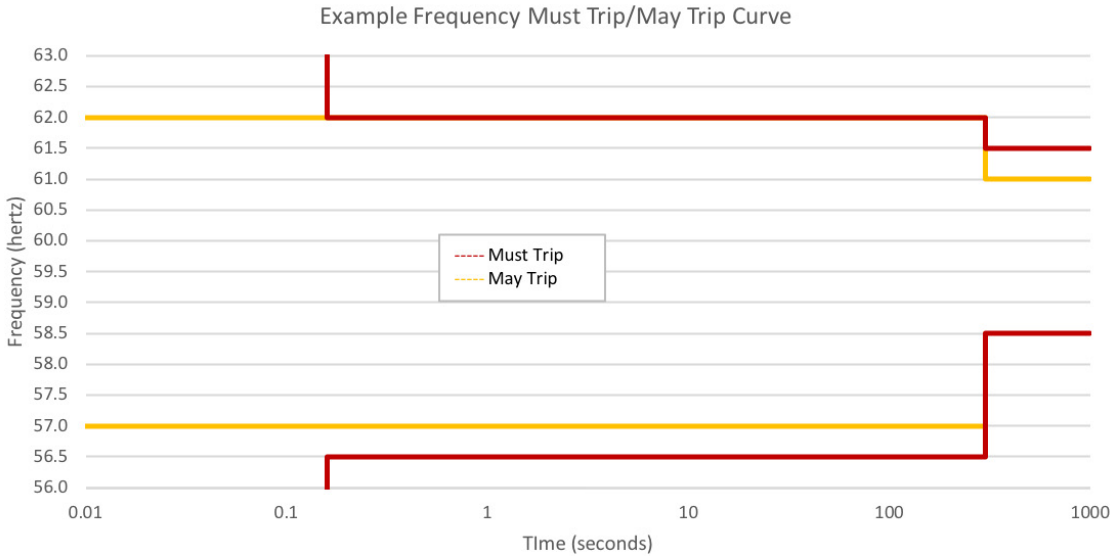


Figure 6—Example low and high-frequency ride-through curves

Table 52—Example low and high-frequency ride-through curve values

Curve	Points
LF must trip	(0.16, 0), (0.16, 56.5), (300, 56.5), (300, 58.5), (1000, 58.5)
LF may trip	(0, 57), (300, 57)
HF must trip	(0.16, 63), (0.16, 62), (300, 62), (300, 61.5), (1000, 61.5)
HF may trip	(0, 62), (300, 62), (300, 61), (1000, 61)

10.10.4.4 DER info resources

10.10.4.4.1 Introduction

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

10.10.4.4.2 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, for instance, the continuous delivered active power rating capability in watts (rtgMaxW), and are available by reading the DERCapability resource.

10.10.4.4.3 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 photovoltaic panels, condition of the equipment, season of the year, or intended use, subject to the maximum limit by rtgMaxW. It should be noted that DERSettings are not likely to be updated by a remote party (such as a utility), but rather locally by, for example, an installer.

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.

10.10.4.4.4 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.

10.10.4.4.5 DERAvailability

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.

10.10.5 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 rtgMaxVar and setMaxVar.

If there is a significant difference between delivered and received var capability, then the DER MAY also include rtgMaxVarNeg and setMaxVarNeg or use rtgMaxVar 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Each unique DER instance SHALL link to a DERCapability instance that SHALL contain type, modesSupported, and rtgMaxW attributes. The type attribute determines which additional modes and attributes are required as shown in Table 53 and Table 54.

Table 53 and Table 54 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 rtgMaxChargeRateW and expose its maximum charging rate via the rtgMaxW attribute. However, if a storage DER also supports discharge mode, then it MUST use rtgMaxChargeRateW to expose its charging rate and MAY use either rtgMaxDischargeRateW (if the DER supports simultaneous generation and discharge) or rtgMaxW to expose its maximum discharging rate.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table 53—Modes and attributes for generator type DERs

Opt1	modesSupported	Opt2	Related attribute(s)	Notes
M	20: Maximum Active Power	M	rtgMaxW/setMaxW	Continuous active power output (includes maximum discharge rate if combined generator/storage type)
		O	rtgMaxVA/setMaxVA	Continuous apparent power out
M	24: Volt-Watt			
R	6: Fixed var or 4 or 5: Fixed PF			
	6: Fixed var	M	rtgMaxVar/setMaxVar	If Fixed Var mode is supported, then Volt-Var mode SHOULD be supported
		O	rtgMaxVarNeg/ setMaxVarNeg	SHOULD be included if received and delivered var differ significantly
	4 or 5: Fixed PF	M	rtgMinPFOverExcited/ setMinPFOverExcited	If Fixed PF mode is supported, then Watt-PF mode SHOULD be supported
		O	rtgMinPFUnderExcited/ setMinPFUnderExcited	SHOULD be included if received and delivered var differ significantly

Table 54—Modes and attributes for storage type DERs

Opt1	modesSupported	Opt2	Related attribute(s)	Notes
M	0: Charge mode	M	rtgMaxChargeRateW/ setMaxChargeRateW or rtgMaxW/setMaxW	rtgMaxChargeRateW is “M” if Discharge mode is supported
		M	rtgMaxWh or rtgMaxAh	Storage capacity
O	1: Discharge mode	M	rtgMaxChargeRateW/ setMaxChargeRateW	
		M	rtgMaxDischargeRateW/ setMaxDischargeRateW or rtgMaxW/setMaxW	rtgMaxDischargeRateW is “M” if combined generator/storage

Storage type DERs include DERType 80 (other storage system), 81, 82, and 83 (combined PV and storage).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.10.6 LogEvents

Table 55—Distributed energy resources LogEvents

LogEvent name	LogEvent code	LogEvent description
DER_FAULT_OVER_CURRENT	0x00	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to overcurrent
DER_FAULT_OVER_CURRENT_RTN	0x01	SHOULD be generated when a fault due to overcurrent clears
DER_FAULT_OVER_VOLTAGE	0x02	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to over voltage
DER_FAULT_OVER_VOLTAGE_RTN	0x03	SHOULD be generated when a fault due to over voltage clears
DER_FAULT_UNDER_VOLTAGE	0x04	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to under voltage
DER_FAULT_UNDER_VOLTAGE_RTN	0x05	SHOULD be generated when a fault due to under voltage clears
DER_FAULT_OVER_FREQUENCY	0x06	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to over frequency
DER_FAULT_OVER_FREQUENCY_RTN	0x07	SHOULD be generated when a fault due to over frequency clears
DER_FAULT_UNDER_FREQUENCY	0x08	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to under frequency
DER_FAULT_UNDER_FREQUENCY_RTN	0x09	SHOULD be generated when a fault due to under frequency clears
DER_FAULT_VOLTAGE_IMBALANCE	0x0A	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to a voltage imbalance
DER_FAULT_VOLTAGE_IMBALANCE_RTN	0x0B	SHOULD be generated when a fault due to a voltage imbalance clears
DER_FAULT_CURRENT_IMBALANCE	0x0C	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to a current imbalance
DER_FAULT_CURRENT_IMBALANCE_RTN	0x0D	SHOULD be generated when a fault due to a current imbalance clears
DER_FAULT_EMERGENCY_LOCAL	0x0E	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to a local emergency
DER_FAULT_EMERGENCY_LOCAL_RTN	0x0F	SHOULD be generated when a fault due to a local emergency clears
DER_FAULT_EMERGENCY_REMOTE	0x10	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to a remote emergency
DER_FAULT_EMERGENCY_REMOTE_RTN	0x11	SHOULD be generated when a fault due to a remote emergency clears
DER_FAULT_LOW_POWER_INPUT	0x12	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to low input power
DER_FAULT_LOW_POWER_INPUT_RTN	0x13	SHOULD be generated when a fault due to low input power clears
DER_FAULT_PHASE_ROTATION	0x14	SHOULD be generated when a fault occurs (requiring an action such as a shutdown or restart) due to phase rotation
DER_FAULT_PHASE_ROTATION_RTN	0x15	SHOULD be generated when a fault due to phase rotation clears

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.11 Metering Mirror function set

10.11.1 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.

10.11.2 List ordering

Table 56—Metering Mirror list ordering

Resource name	Primary key	Secondary key	Tertiary key
MirrorUsagePoint	mRID (descending)	N/A	N/A

10.11.3 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.

a) To create a new Metering Mirror the client SHALL POST to the server's MirrorUsagePointList (e.g., /mup) for the mirrored usage point.

   1) This POST SHALL contain at least the information through the definition of MirrorMeterReadings and ReadingType including the MirrorUsagePoint mRID and MirrorMeterReading mRIDs.

   2) The POST MAY also contain MirrorReadingSets and Readings.

   3) 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.

   4) 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 Rule h) and Rule i) are to be applied.

b) 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.

c) 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).

d) 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).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

e) The Metering Mirror server SHOULD only accept POSTs to a given MirrorUsagePoint from the client that created the mirror.

f) 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).

g) 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).

h) 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.

   1) In a POST to the MirrorUsagePoint, if the mRID attribute matches a previous MirrorMeterReading, then each of the contained MirrorReadingSets SHALL be handled as required by Rule i) below. The contents of the MirrorMeterReading SHALL overwrite the data in the associated MeterReading.

   2) In a POST to the MirrorUsagePoint, if the mRID does not match a previous MirrorMeterReading and it contains a ReadingType, then a new MeterReading SHALL be created under the associated UsagePoint with the new data.

   3) 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).

i) 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:

   1) 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.

   2) If a MirrorReadingSet is received with a unique mRID, then the new data SHALL be added to the identified ReadingSetList.

j) 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).

k) The Metering Mirror server MAY decide when to remove data from the related UsagePoint resource structure.

l) When a MirrorUsagePoint is deleted, the associated UsagePoint SHALL also be deleted.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

10.11.4 LogEvents

Table 57—Metering mirror LogEvents

LogEvent name	LogEvent code	LogEvent description
MUP_MIRROR_EXPIRED	0x00	SHOULD be generated by a server when a Metering Mirror expires

11. Manufacturer-specific proprietary extensions

11.1 Overview

This clause describes rules and mechanisms for interested parties to extend IEEE Std 2030.5 with proprietary extensions.

It should be noted that as IEEE Std 2030.5 is intended to run over an IP stack, many techniques already exist for providing proprietary services over such a stack with various protocols. This clause is intended for guidance to developers of proprietary extensions who want a design similar to IEEE Std 2030.5, or wish to add extended elements to IEEE Std 2030.5.

11.2 xmDNS/DNS-SD

Proprietary extensions SHALL NOT be made using the "`smartenergy`" Service Type.

It is RECOMMENDED that proprietary extensions that wish to use xmDNS/DNS-SD apply for a new Service Type with which to operate.

11.3 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 standard and in the supplemental material of IEEE Std 2030.5.

11.4 Resources

Proprietary extensions SHALL NOT place any objects, elements, attributes, etc. in the standardized SEP XML namespace ("`urn:ieee:std:2030.5:ns`") 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 in Figure 7 demonstrates allowed and disallowed (crossed out) element extensions.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Allowed

```
<SEPElement1 xmlns="urn:ieee:std:2030.5:ns"
xmlns:MFENS="http://foo.org/mfe">
  <SEPElement2>a</SEPElement2>
  <SEPElement3>b</SEPElement3>
  <MFENS:MFEElement1>c</MFENS:MFEElement1>
</SEPElement1>
```

Disallowed

```
<SEPElement1 xmlns="urn:ieee:std:2030.5:ns">
  <SEPElement2>a</SEPElement2>
  <SEPElement3>b</SEPElement3>
  <MFEElement1>c</MFEElement1>
</SEPElement1>
```

Figure 7—Allowed and disallowed extension

Proprietary extensions SHALL NOT extend enumerations defined in the IEEE 2030.5 schema.

Proprietary extensions made to standardized objects (in a proprietary namespace) defined in the IEEE 2030.5 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 IEEE 2030.5 schema SHALL NOT change the semantics of elements and attributes defined in the IEEE 2030.5 schema.

11.5 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Annex A

(informative)

Web-application description language (WADL)

A.1 Introduction

Note that the WADL (sep_wadl.xml, contained in the supplemental material of IEEE Std 2030.5) is NORMATIVE. This annex presents a human-friendly view of the information to facilitate understanding. While the following annex may contain normative statements, these statements are a human-friendly view of the normative statements in the WADL. Readers should ignore the normative statements in this annex and instead refer to the corresponding normative material in the WADL.

A.2 Support resources section

A.2.1 Device Capability function set

A.2.1.1 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

A.2.2 Self Device Resource function set

A.2.2.1 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

A.2.3 End Device Resource function set

A.2.3.1 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.2.3.2 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

A.2.3.3 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

A.2.3.4 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

A.2.4 Function Set Assignments function set

A.2.4.1 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

A.2.4.2 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

A.2.5 Subscription/Notification Mechanism function set

A.2.5.1 SubscriptionList resource

Contains Subscriptions related to the indicated device. Documented in Subscription/Notification Mechanism.

Copyright © 2018 IEEE. All rights reserved.

Sample URI: /edev/{id1}/sub
Request representation: Subscription
Response representation: SubscriptionList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

A.2.5.2 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

A.2.5.3 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

A.2.5.4 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

A.2.6 Response function set

A.2.6.1 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

A.2.6.2 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

A.2.6.3 ResponseList resource

List of Response instances.
Sample URI: /rsps/{id1}/rsp
Request representation: Response

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Response representation: ResponseList
Methods: GET/HEAD: Optional, PUT: Error, POST: Mandatory, DELETE: Error

A.2.6.4 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

A.2.6.5 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

A.2.6.6 TextResponse resource

A specific TextMessage Response instance.
Sample URI: /rsps/{id1}/rsp/{id2}
Request representation: TextResponse
Response representation: TextResponse
Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

A.2.6.7 DERControlResponse resource

A specific DERControl Response instance.
Sample URI: /rsps/{id1}/rsp/{id2}
Request representation: DERControlResponse
Response representation: DERControlResponse
Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

A.2.6.8 FlowReservationResponseResponse resource

A specific FlowReservationResponse Response instance.
Sample URI: /rsps/{id1}/rsp/{id2}
Request representation: FlowReservationResponseResponse
Response representation: FlowReservationResponseResponse
Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

A.2.6.9 DrResponse resource

A specific Demand Response/Load Control EndDeviceControl Response (DrResponse) instance.
Sample URI: /rsps/{id1}/rsp/{id2}
Request representation: DrResponse
Response representation: DrResponse
Methods: GET/HEAD: Optional, PUT: Error, POST: Error, DELETE: Optional

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.3 Common resources section

A.3.1 Time function set

A.3.1.1 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

A.3.2 Device Information function set

A.3.2.1 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

A.3.2.2 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

A.3.2.3 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

A.3.3 Power Status function set

A.3.3.1 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.3.4 Network Status function set

A.3.4.1 IPInterfaceList resource

List of IPInterface instances on the associated EndDevice or SelfDevice.
Sample URI: /edev/{id1}/ns
Request representation: IPInterface
Response representation: IPInterfaceList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

A.3.4.2 IPInterface resource

Specific IPInterface resource. This resource may be thought of as network status information for a specific network (Internet Protocol [IP]) layer interface.
Sample URI: /edev/{id1}/ns/{id2}
Request representation: IPInterface
Response representation: IPInterface
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

A.3.4.3 IPAddrList resource

List of IP addresses.
Sample URI: /edev/{id1}/ns/{id2}/addr
Request representation: IPAddr
Response representation: IPAddrList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

A.3.4.4 IPAddr resource

A specific IP address.
Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}
Request representation: IPAddr
Response representation: IPAddr
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

A.3.4.5 RPLInstanceList resource

List of RPL instances that the IPAddr is a member.
Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl
Request representation: RPLInstance
Response representation: RPLInstanceList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

A.3.4.6 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: Optional, POST: Error, DELETE: Optional

A.3.4.7 RPLSourceRoutesList resource

List of RPL source routes.
Sample URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt

Copyright © 2018 IEEE. All rights reserved.

Request representation: RPLSourceRoutes
Response representation: RPLSourceRoutesList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

A.3.4.8 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: Optional, POST: Error, DELETE: Optional

A.3.4.9 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

A.3.4.10 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: Optional, POST: Error, DELETE: Optional

A.3.4.11 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

A.3.4.12 Neighbor resource

A specific 802.15.4 neighbor.
Sample URI: /edev/{id1}/ns/{id2}/ll/{id3}/nbh/{id4}
Request representation: Neighbor
Response representation: Neighbor
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

A.3.5 Log Event function set

A.3.5.1 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.3.5.2 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

A.4 Smart energy resources section

A.4.1 Configuration Resource function set

A.4.1.1 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

A.4.1.2 PriceResponseCfgList resource

Contains a list of price-response configuration settings.
Sample URI: /edev/{id1}/cfg/prcfg
Request representation: PriceResponseCfg
Response representation: PriceResponseCfgList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

A.4.1.3 PriceResponseCfg resource

Contains the price-response configuration settings for this EndDevice associated with a RateComponent.
Sample URI: /edev/{id1}/cfg/prcfg/{id2}
Request representation: PriceResponseCfg
Response representation: PriceResponseCfg
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Mandatory

A.4.2 Software Download function set

A.4.2.1 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

A.4.2.2 File resource

A specific file.
Sample URI: /file/{id1}

Copyright © 2018 IEEE. All rights reserved.

Request representation: File
Response representation: File
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Discouraged

A.4.2.3 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

A.4.3 Demand Response and Load Control function set

A.4.3.1 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

A.4.3.2 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

A.4.3.3 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

A.4.3.4 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

A.4.3.5 EndDeviceControl resource

Specific EndDeviceControl resource. This resource can be thought of as a particular Demand Response/Load Control event for a period of time.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Sample URI: /dr/{id1}/edc/{id2}
Request representation: EndDeviceControl
Response representation: EndDeviceControl
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

A.4.3.6 LoadShedAvailabilityList resource

A list of LoadShedAvailability instances.
Sample URI: /edev/{id1}/lsl
Request representation: LoadShedAvailability
Response representation: LoadShedAvailabilityList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Mandatory, DELETE: Error

A.4.3.7 LoadShedAvailability resource

A specific LoadShedAvailability entry from the LoadShedAvailabilityList.
Sample URI: /edev/{id1}/lsl/{id2}
Request representation: LoadShedAvailability
Response representation: LoadShedAvailability
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Mandatory

A.4.4 Metering function set

A.4.4.1 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

A.4.4.2 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: Error, DELETE: Optional

A.4.4.3 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

A.4.4.4 MeterReading resource

Meter reading instance which contains ReadingSet and Reading resources.
Sample URI: /upt/{id1}/mr/{id2}
Request representation: MeterReading

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Response representation: MeterReading
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Optional

A.4.4.5 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

A.4.4.6 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

A.4.4.7 ReadingSet resource

ReadingSet 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

A.4.4.8 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

A.4.4.9 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

A.4.4.10 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

A.4.4.11 MirrorUsagePoint resource

Mirror usage point instance including resources it supports.
Sample URI: /mup/{id1}

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Request representation: MirrorUsagePoint (PUT), MirrorMeterReading (POST), MirrorMeterReadingList (POST)
Response representation: MirrorUsagePoint
Methods: GET/HEAD: Optional, PUT: Mandatory, POST: Mandatory, DELETE: Mandatory

A.4.5 Pricing function set

A.4.5.1 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

A.4.5.2 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

A.4.5.3 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

A.4.5.4 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

A.4.5.5 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

A.4.5.6 TimeTariffIntervalList resource

Collection of TimeTariffInterval instances, including associated ConsumptionTariffInterval.
Sample URI: /tp/{id1}/rc/{id2}/tti

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Request representation: TimeTariffInterval
Response representation: TimeTariffIntervalList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

A.4.5.7 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

A.4.5.8 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

A.4.5.9 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

A.4.6 Messaging function set

A.4.6.1 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

A.4.6.2 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

A.4.6.3 ActiveTextMessageList resource

List of Messages that are currently active.
Sample URI: /msg/{id1}/acttxt

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Request representation: TextMessage
Response representation: TextMessageList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Error

A.4.6.4 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

A.4.6.5 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

A.4.7 Billing function set

A.4.7.1 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

A.4.7.2 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

A.4.7.3 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

A.4.7.4 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.4.7.5 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

A.4.7.6 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

A.4.7.7 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

A.4.7.8 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

A.4.7.9 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

A.4.7.10 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

A.4.7.11 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.4.7.12 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

A.4.7.13 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

A.4.7.14 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

A.4.7.15 TargetReading resource

A specific target reading channel.
Sample URI: /bill/{id1}/ca/{id2}/tar/{id3}
Request representation: TargetReading
Response representation: TargetReading
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

A.4.7.16 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: Discouraged, DELETE: Error

A.4.7.17 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

A.4.7.18 ServiceSupplier resource

A specific service supplier.
Sample URI: /bill/{id1}/ss
Request representation: ServiceSupplier
Response representation: ServiceSupplier
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Error, DELETE: Discouraged

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.4.8 Prepayment function set

A.4.8.1 PrepaymentList resource

A list of Prepayment instances.
Sample URI: /ppy
Request representation: Prepayment
Response representation: PrepaymentList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

A.4.8.2 Prepayment resource

A particular Prepayment instance. Provides links to the AccountBalance, CreditRegister, and OperationStatus 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

A.4.8.3 AccountBalance resource

AccountBalance instance.
Sample URI: /ppy/{id1}/ab
Request representation: AccountBalance
Response representation: AccountBalance
Methods: GET/HEAD: Mandatory, PUT: Discouraged, POST: Error, DELETE: Error

A.4.8.4 PrepayOperationStatus resource

The OperationStatus 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

A.4.8.5 ActiveSupplyInterruptionOverrideList 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

A.4.8.6 SupplyInterruptionOverrideList resource

A list of SupplyInterruptionOverride instances.
Sample URI: /ppy/{id1}/si
Request representation: SupplyInterruptionOverride
Response representation: SupplyInterruptionOverrideList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Discouraged, DELETE: Error

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.4.8.7 SupplyInterruptionOverride resource

A particular SupplyInterruptionOverride 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

A.4.8.8 CreditRegisterList resource

A list of CreditRegister 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

A.4.8.9 CreditRegister resource

A particular CreditRegister 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

A.4.9 Flow Reservation function set

A.4.9.1 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

A.4.9.2 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

A.4.9.3 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.4.9.4 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

A.4.10 Distributed Energy Resources function set

A.4.10.1 DERList resource

A list of distributed energy resources (DERs).
Sample URI: /edev/{id1}/der
Request representation: DER
Response representation: DERList
Methods: GET/HEAD: Mandatory, PUT: Error, POST: Optional, DELETE: Error

A.4.10.2 DER resource

The information about a specific distributed energy resource.
Sample URI: /edev/{id1}/der/{id2}
Request representation: DER
Response representation: DER
Methods: GET/HEAD: Mandatory, PUT: Optional, POST: Error, DELETE: Optional

A.4.10.3 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

A.4.10.4 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: Optional, DELETE: Error

A.4.10.5 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

A.4.10.6 DERSettings resource

The DER settings of the associated EndDevice or SelfDevice.

Copyright © 2018 IEEE. All rights reserved.

Sample URI: /edev/{id1}/der/{id2}/derg
Request representation: DERSettings
Response representation: DERSettings
Methods: GET/HEAD: Mandatory, PUT: Mandatory, POST: Error, DELETE: Error

A.4.10.7 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

A.4.10.8 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

A.4.10.9 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

A.4.10.10 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

A.4.10.11 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

A.4.10.12 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

A.4.10.13 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

A.4.10.14 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

A.4.10.15 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

A.4.10.16 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

A.4.10.17 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Annex B

(informative)

IEEE 2030.5 model

B.1 Introduction

Note that the XML Schema (sep.xsd, contained in the supplemental material of IEEE Std 2030.5) is NORMATIVE. This annex presents a human-friendly view of the information to facilitate understanding. While the following annex may contain normative statements, these statements are a human-friendly view of the normative statements in the XML Schema. Readers should ignore the normative statements in this annex and instead refer to the corresponding normative material in the XML Schema.

B.2 IEEE 2030.5 package

B.2.1 Introduction

The IEEE 2030.5 model is organized into function sets, represented by sub-packages. However, all structures are defined inside a single namespace.

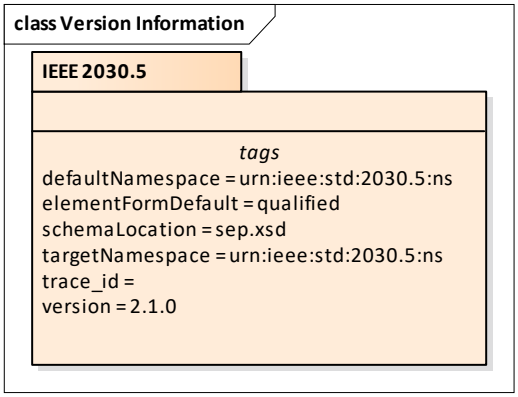


Figure B.1—Version information

B.2.2 DeviceCapability package

Contains definition of the objects used to convey the resources that are implemented by the publishing host.

Copyright © 2018 IEEE. All rights reserved.

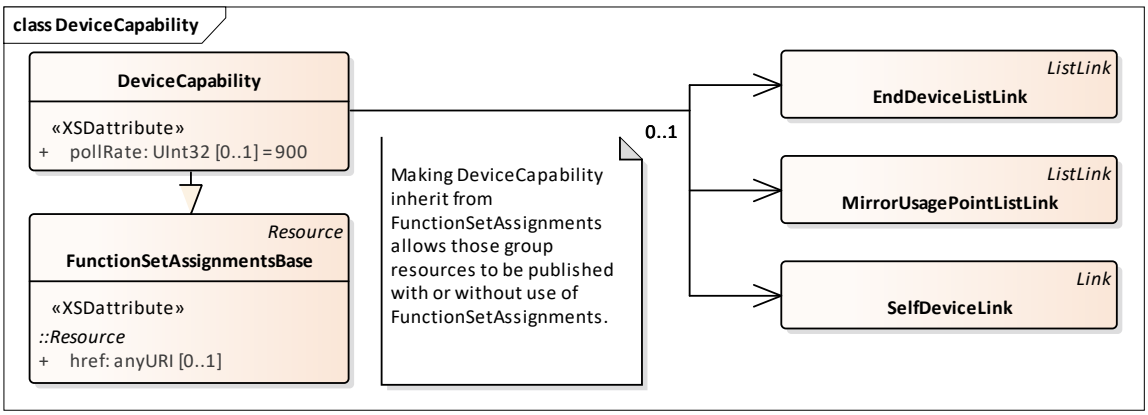


Figure B.2—DeviceCapability

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.

pollRate attribute *(UInt32) [0..1] «XSDattribute»*
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

B.2.3 Common package

B.2.3.1 Introduction

This package contains objects that are used in multiple function sets.

B.2.3.2 Identification package

Contains super-classes that define the attributes common to categories of objects.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

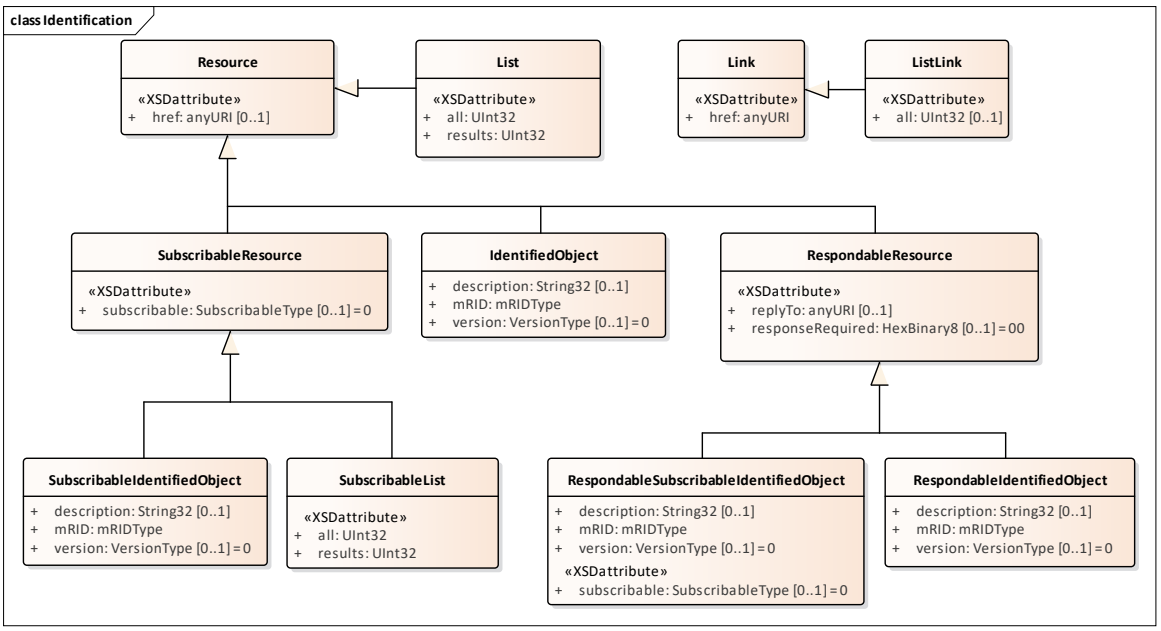


Figure B.3—Identification

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 Clause 4 (design pattern) for additional details.

all attribute (UInt32) «XSDattribute»

The number specifying “all” of the items in the list. Required on a response to a GET, ignored otherwise.

results attribute (UInt32) «XSDattribute»

Indicates the number of items in this page of results.

ListLink Object (Link)

ListLinks provide a reference, via URI, to a List.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

all attribute (UInt32) [0..1] «XSDattribute»
Indicates the total number of items in the referenced list. This attribute SHALL be present if the href is a local or relative URI. This attribute SHOULD NOT be present if the href is a remote or absolute URI, as the server may be unaware of changes to the value.

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 27 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.

Copyright © 2018 IEEE. All rights reserved.

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 (UInt32) «XSDattribute»
The number specifying "all" of the items in the list. Required on GET, ignored otherwise.

results attribute (UInt32) «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).

B.2.3.3 Objects package

Contains definitions of objects used by multiple function sets.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

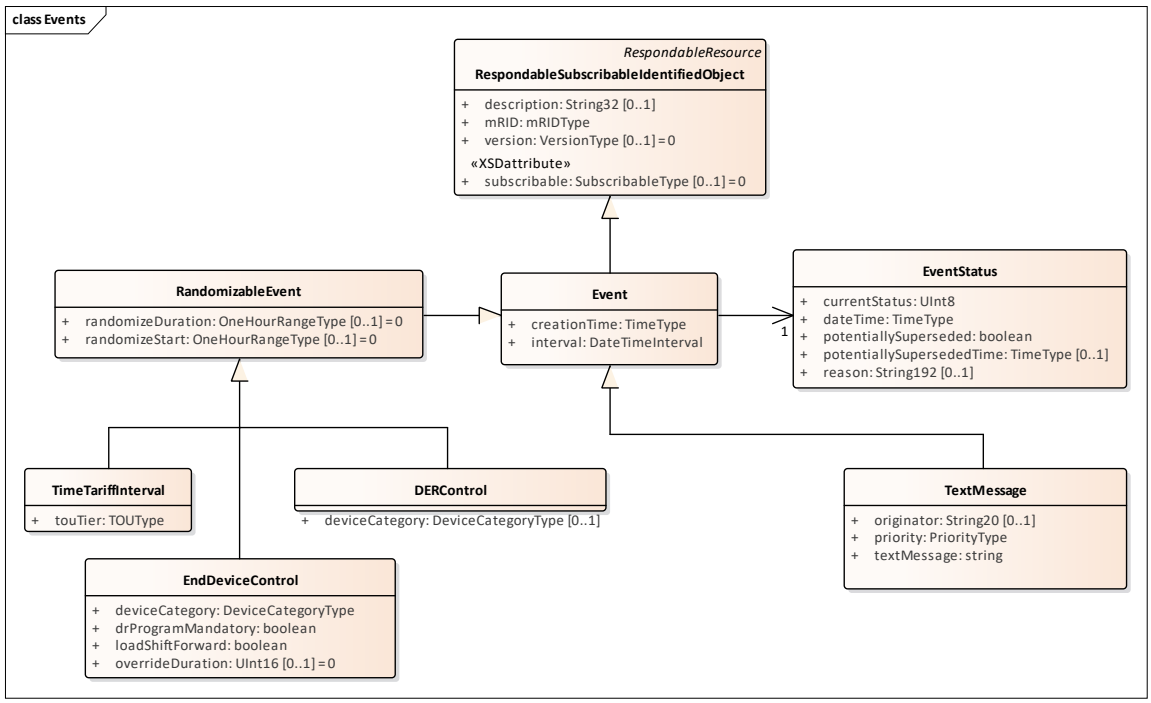


Figure B.4—Events

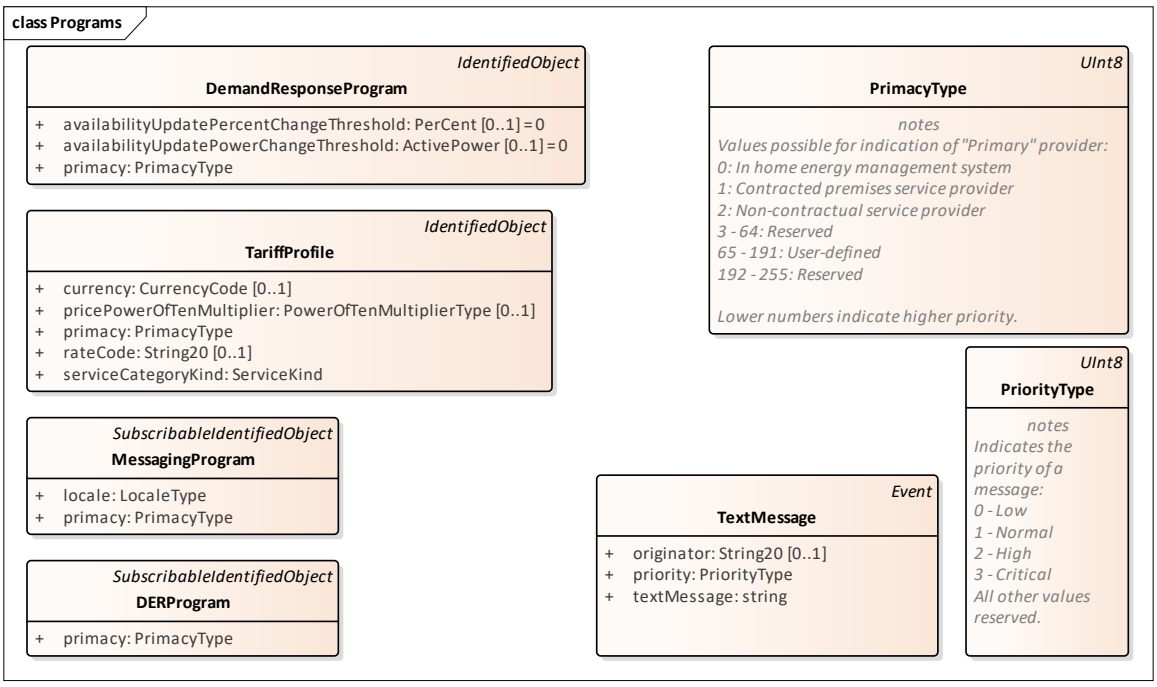


Figure B.5—Programs

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

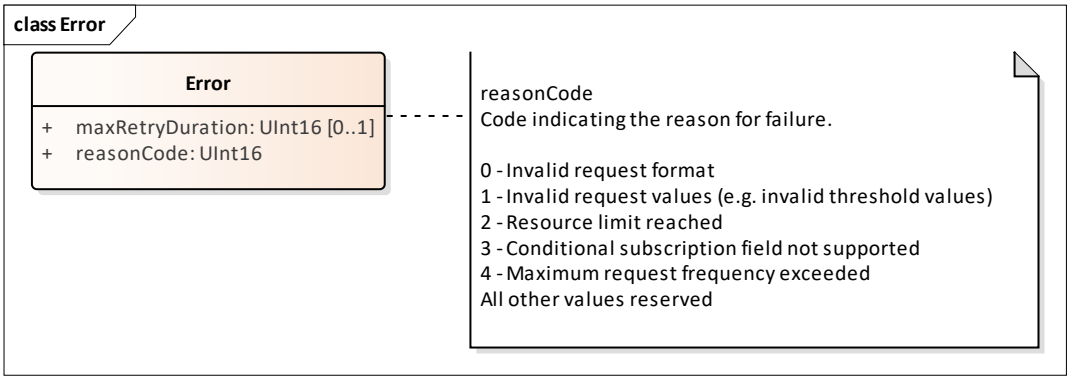


Figure B.6—Error

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, 9.2.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 from the same program that targets the exact same set of deviceCategory's (if applicable) and DERControl controls (e.g., opModTargetW) (if applicable), 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 overlap this event in time and also overlap in some, but not all, deviceCategory's (if applicable) and DERControl controls (e.g., opModTargetW) (if applicable) in the same function set instance.

Copyright © 2018 IEEE. All rights reserved.

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.

B.2.3.4 Types package

Contains definitions of reusable data types.

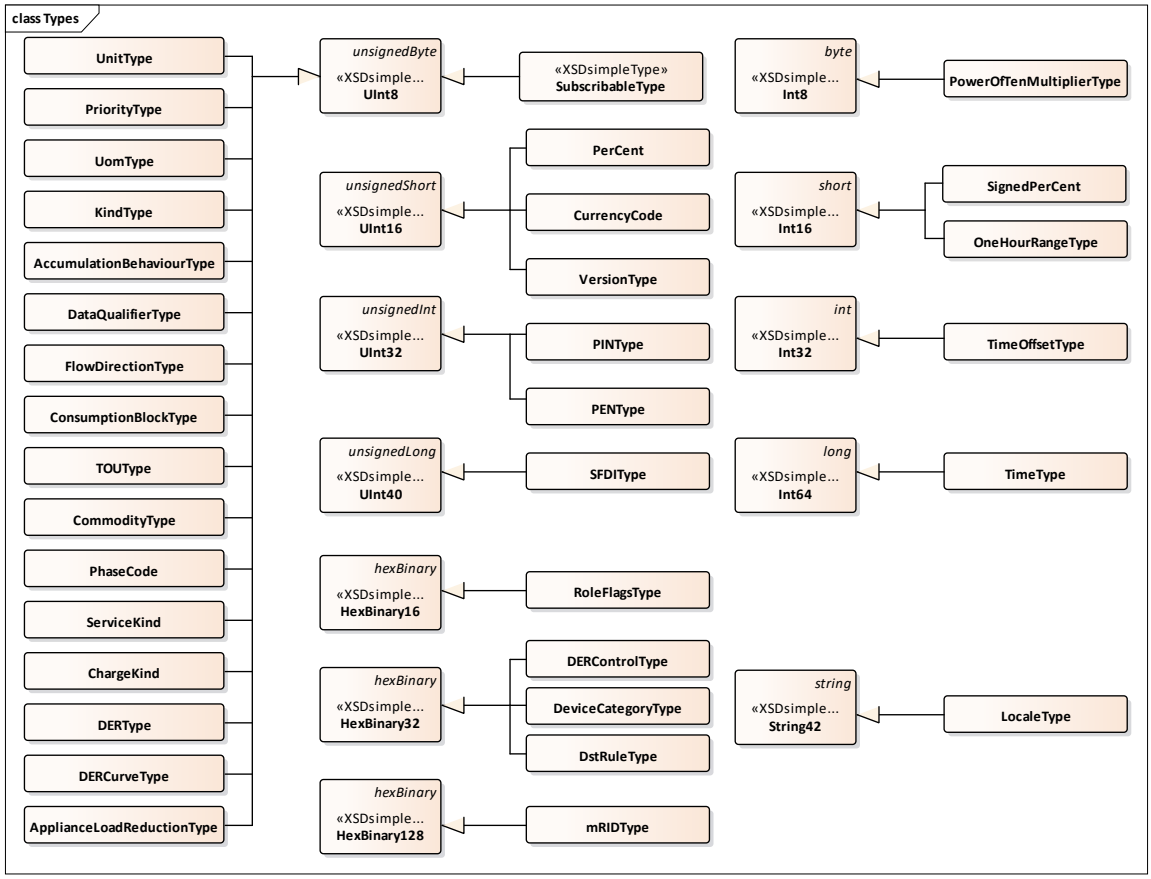


Figure B.7—Types

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Copyright © 2018 IEEE. All rights reserved.

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 on the low voltage, or secondary, side of a service transformer)

2 = Electricity primary metered value (measured on the high voltage, or primary, side of the service transformer)

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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

11 = Block 11

12 = Block 12

13 = Block 13

14 = Block 14

15 = Block 15

16 = Block 16

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 = U.S. 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

29 = Standard deviation of a population (typically indicated by a lower case sigma)

30 = Standard deviation of a sample drawn from a population (typically indicated by a lower case “s”)

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 = Load control switch

14 = Energy management system

15 = Smart energy module

16 = Electric vehicle

17 = Electric vehicle supply equipment (EVSE)

18 = Virtual or mixed DER

19 = Reciprocating engine

20 = Fuel cell

21 = Photovoltaic system

22 = Combined heat and power

23 = Combined PV and storage

24 = Other generation system

Copyright © 2018 IEEE. All rights reserved.

25 = Other storage system

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 to 11: seconds 0 to 3599

Bits 12 to 16: hours 0 to 23

Bits 17 to 19: day of the week 0 = not applicable, 1 to 7 (Monday = 1)

Bits 20 to 24: day of the month 0 = not applicable, 1 to 31

Bits 25 to 27: operator (detailed below)

Bits 28 to 31: month 1 to 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 the third Friday in March at 1:45 AM. The rule...

Seconds: 2700

Hours: 1

Day of week: 5

Day of month: 0

Operator: 4

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

GPSLocationType object «Compound» ()
Specifies a GPS location, expressed in WGS 84 coordinates.

lat attribute (String32)
Specifies the latitude from equator. −90 (south) to +90 (north) in decimal degrees.

lon attribute (String32)
Specifies the longitude from Greenwich Meridian. −180 (west) to +180 (east) in decimal degrees.

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)
IETF RFC 4646 identifier of a language-region.

mRIDType object (HexBinary128)
A master resource identifier. The IANA Private Enterprise Number (PEN) provider ID SHALL be specified in bits 0 to 31, the least-significant bits, and objects created by that provider SHALL be assigned unique IDs with the remaining 96 bits.

0xFFFFFFFFFFFFFFFFFFFFFFFF[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 to 10 000. (10 000 = 100%)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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)

Six digit unsigned decimal integer (0 to 999999).

(Note that this only requires 20 bits, if it can be allocated.)

PowerOfTenMultiplierType object (Int8)

−9 = Nano $= \times 10^{-9}$

−6 = Micro $= \times 10^{-6}$

−3 = Milli $= \times 10^{-3}$

0 = None $= = \times 1$ (default, if not specified)

1 = Deca $= \times 10$

2 = Hecto $= \times 100$

3 = Kilo $= \times 1000$

6 = Mega $= \times 10^{6}$

9 = Giga $= \times 10^{9}$

This is not a complete list. Any integer between −9 and 9 SHALL be supported, indicating the power of ten multiplier for the units.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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

3 to 64 -= Reserved

65 to 191 = User-defined

192 to 255 = Reserved

Lower numbers indicate higher priority.

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 to 15—Reserved

ServiceKind object (UInt8)
Service kind:

0 = Electricity

1 = Gas

2 = Water

Copyright © 2018 IEEE. All rights reserved.

3 = Time

4 = Pressure

5 = Heat

6 = Cooling

All other values reserved.

SFDIType object (UInt40)
Unsigned integer, maximum inclusive 687194767359, which is $2^{36} - 1$ (68 719 476 735), with added check digit. See 8.4.2 for check digit calculation.

SignedPerCent object (Int16)
Used for signed percentages, specified in hundredths of a percent, −10 000 to 10 000. (10 000 = 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 1 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 = U.S. gallons

6 = Imperial gallons

7 = British thermal units (BTUs)

8 = Liters

9 = kPA (gauge)

10 = kPA (absolute)

11 = Megajoule

12 = Unitless

All other values reserved.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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)

The following values are recommended values sourced from the unit of measure enumeration in IEC 61968-9. Other values from the unit of measure enumeration in IEC 61968-9 MAY be used.

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 = m^3 (cubic meter)

61 = VA (apparent power)

63 = var (reactive power)

65 = CosTheta (displacement power factor)

67 = V^2 (volts squared)

69 = A^2 (amp squared)

71 = VAh (apparent energy)

72 = Wh (real energy in watt-hours)

73 = varh (reactive energy)

106 = Ah (ampere-hours/available charge)

119 = ft^3 (cubic feet)

122 = ft^3/h (cubic feet per hour)

Copyright © 2018 IEEE. All rights reserved.

125 = m^3/h (cubic meter per hour)

128 = U.S. gallons

129 = U.S. gal/h (U.S. gallons per hour)

130 = Imperial gallons

131 = IMP gal/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

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.

B.2.3.5 Primitive types package

Contains definitions of primitive data types based on XML Schema primitives.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

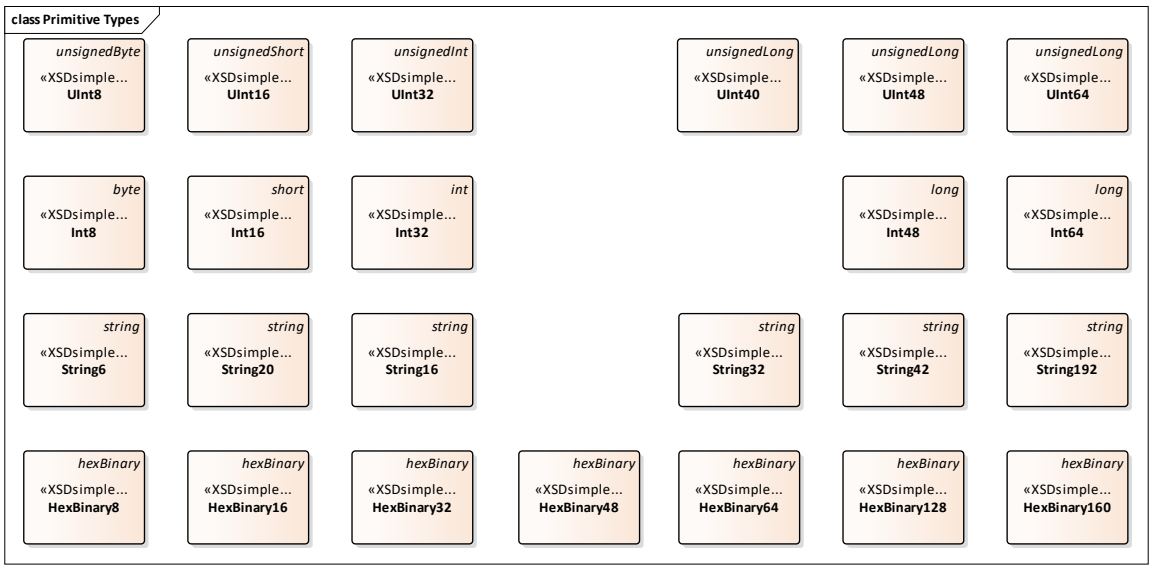


Figure B.8—Primitive types

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 maximum). 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 maximum). 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 maximum). 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 maximum). 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 maximum). 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 maximum). 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”.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

String6 object «XSDsimpleType» (string)
Character string of maximum 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 maximum 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 maximum 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 maximum 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 maximum 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 maximum 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, maximum inclusive 255 ($2^8 - 1$).

UInt16 object «XSDsimpleType» (unsignedShort)
Unsigned integer, maximum inclusive 65 535 ($2^{16} - 1$).

UInt32 object «XSDsimpleType» (unsignedInt)
Unsigned integer, maximum inclusive 4 294 967 295 ($2^{32} - 1$).

UInt40 object «XSDsimpleType» (unsignedLong)
Unsigned integer, maximum inclusive 1 099 511 627 775 ($2^{40} - 1$).

UInt48 object «XSDsimpleType» (unsignedLong)
Unsigned integer, maximum inclusive 281 474 976 710 655 ($2^{48} - 1$).

UInt64 object «XSDsimpleType» (unsignedLong)
Unsigned integer, maximum inclusive 18 446 744 073 709 551 615 ($2^{64} - 1$).

Int8 object «XSDsimpleType» (byte)
Signed integer, minimum −128, maximum 127.

Int16 object «XSDsimpleType» (short)
Signed integer, minimum −32 768, maximum 32 767.

Copyright © 2018 IEEE. All rights reserved.

Int32 object «XSDsimpleType» (int)

Signed integer, maximum inclusive 2 147 483 647 (2^{31}), minimum inclusive −2 147 483 647 (same as xs:int).

Int48 object «XSDsimpleType» (long)

Signed integer, maximum inclusive 140 737 488 355 328 (2^{47}), minimum inclusive −140 737 488 355 328.

Int64 object «XSDsimpleType» (long)

Signed integer, maximum inclusive 9 223 372 036 854 775 807 (2^{63}), minimum inclusive −9 223 372 036 854 775 808 (same as xs:long).

B.2.4 EndDevice package

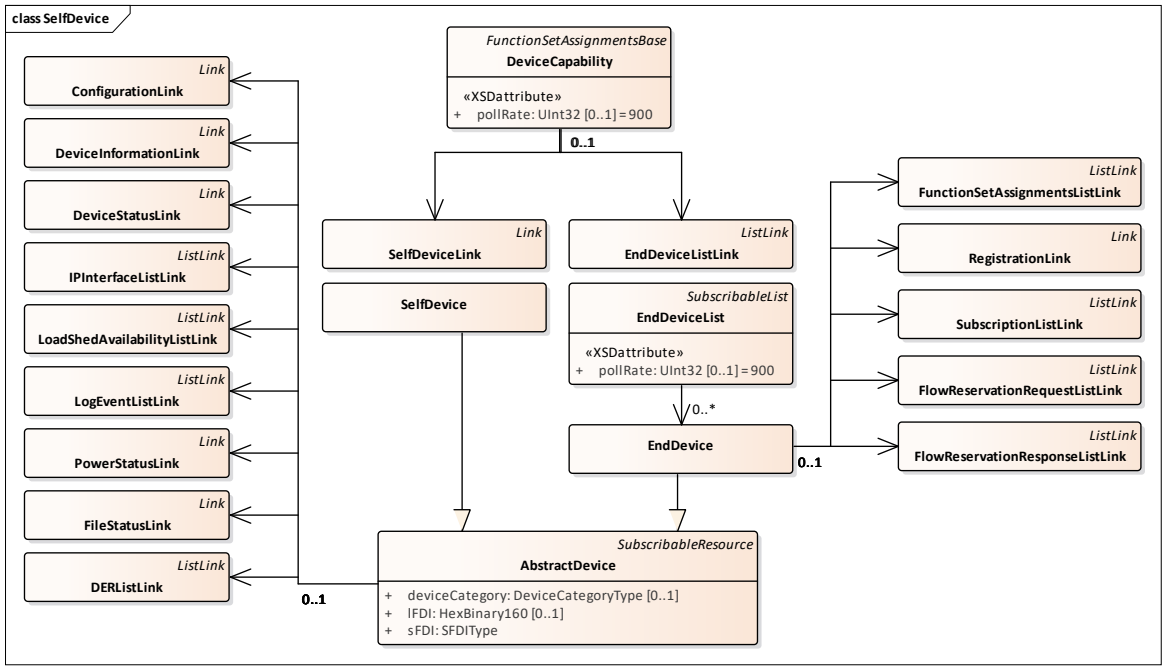


Figure B.9—SelfDevice

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

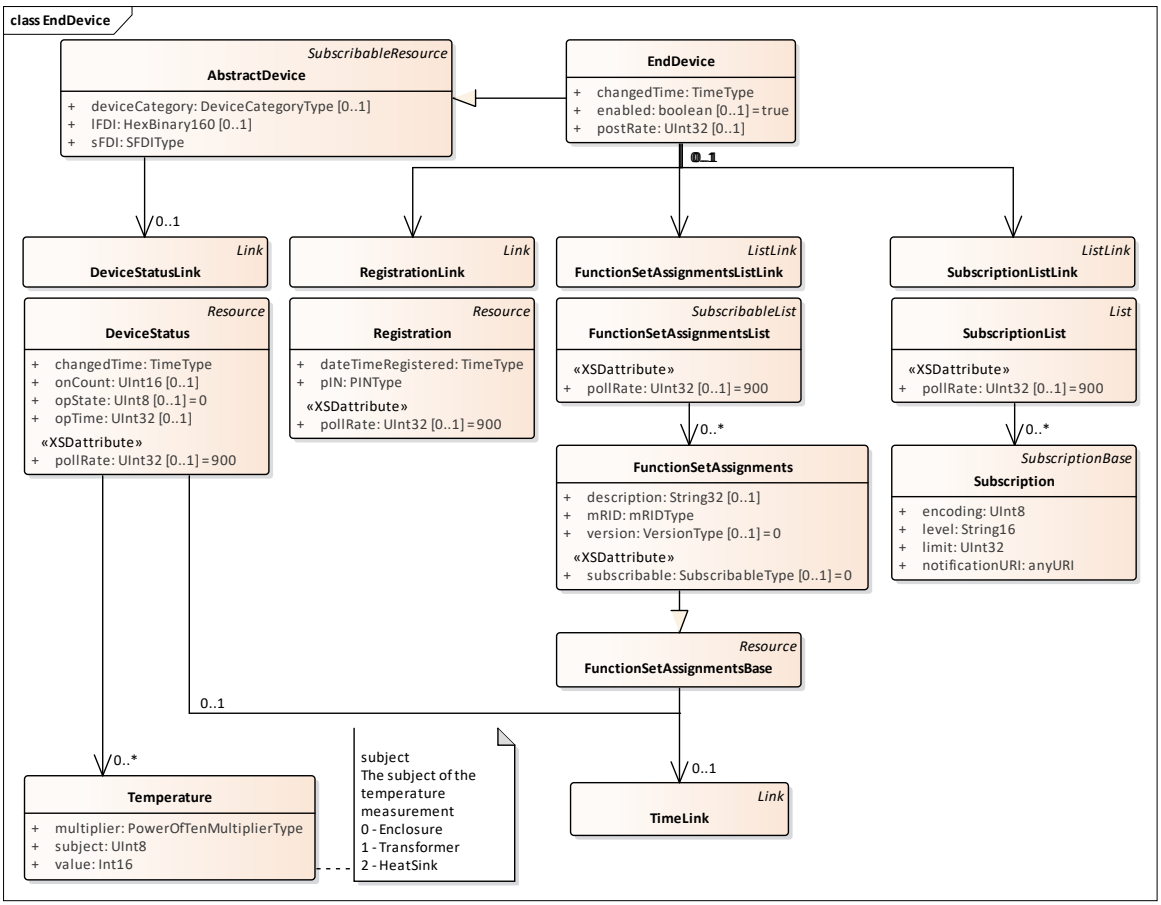


Figure B.10—EndDevice

AbstractDevice object (SubscribableResource)

The EndDevice providing the resources available within the DeviceCapabilities.

deviceCategory attribute (DeviceCategoryType) [0..1]

This field is for use in devices that can adjust energy usage (e.g., demand response, distributed energy resources). For devices that do not respond to EndDeviceControls or DERControls (for instance, an energy services interface [ESI]), this field should not have any bits set.

lFDI attribute (HexBinary160) [0..1]

Long form of device identifier. See Clause 6 (security) for additional details.

sFDI attribute (SFDIType)

Short form of device identifier, WITH the checksum digit. See Clause 6 (security) 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

opState attribute (UInt8) [0..1]
Device operational state:

0 = Not applicable/Unknown

1 = Not operating

2 = Operating

3 = Starting up

4 = Shutting 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.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

EndDevice object (AbstractDevice)
Asset container that performs one or more end device functions. Contains information about individual devices in the network.

changedTime attribute (TimeType)
The time at which this resource was last modified or created.

enabled attribute (boolean) [0..1]
This attribute indicates whether or not an EndDevice is enabled, or registered, on the server. If a server sets this attribute to false, the device is no longer registered. It should be noted that servers can delete EndDevice instances, but using this attribute for some time is more convenient for clients.

postRate attribute (UInt32) [0..1]
POST rate, or how often EndDevice and subordinate resources should be POSTed, in seconds. A client MAY indicate a preferred postRate when POSTing EndDevice. A server MAY add or modify postRate to indicate its preferred posting rate.

EndDeviceList object (SubscribableList)
A List element to hold EndDevice objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

Registration object (Resource)
Registration represents an authorization to access the resources on a host.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 personal identification number (PIN) associated with the device, including the checksum digit.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

SelfDevice object (AbstractDevice)
The EndDevice providing the resources available within the DeviceCapabilities.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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).

Copyright © 2018 IEEE. All rights reserved.

B.2.5 FunctionSetAssignments package

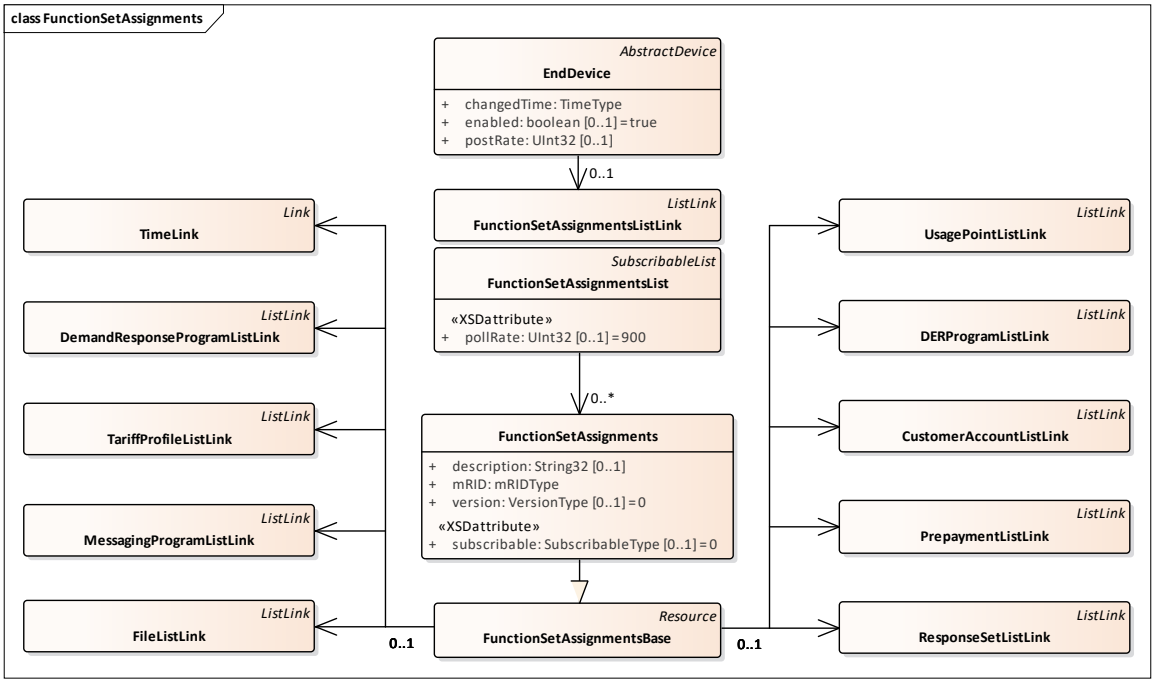


Figure B.11—FunctionSetAssignments

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.

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.

FunctionSetAssignmentsList object (SubscribableList)

A List element to hold FunctionSetAssignments objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

B.2.6 Pub-Sub package

Contains resource definitions used to allow subscriptions and notifications of publications.

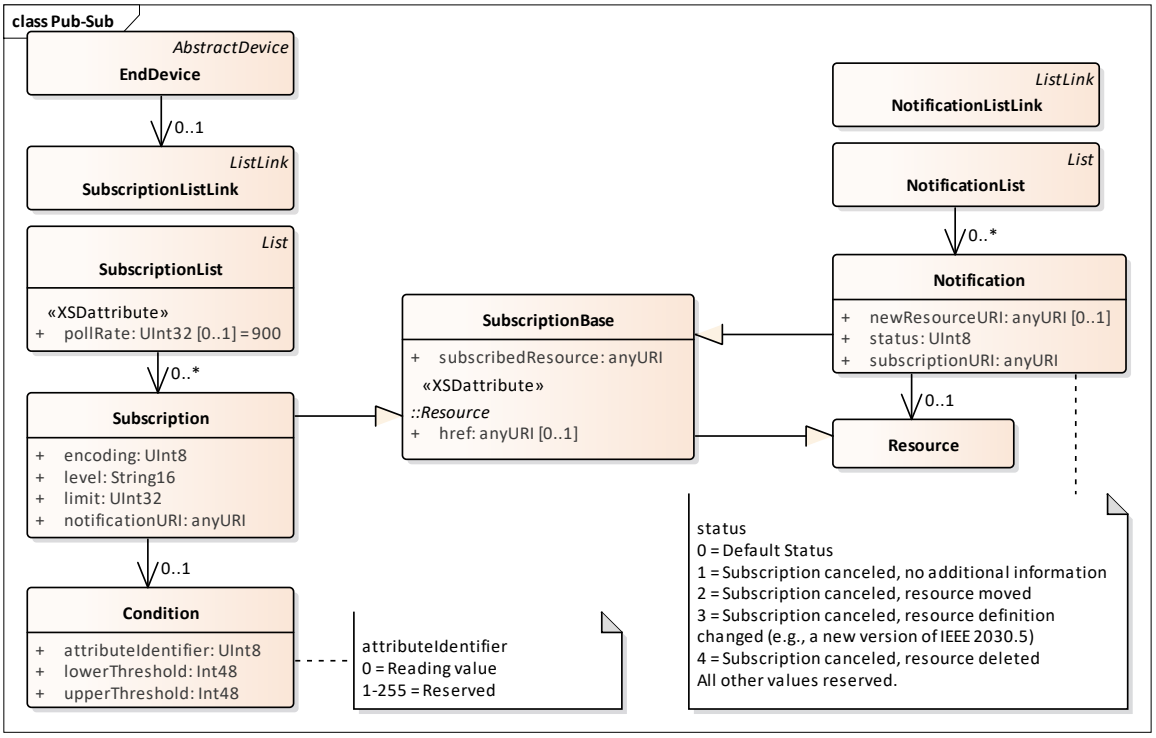


Figure B.12—Pub-Sub

Condition object ()

Indicates a condition that must be satisfied for the Notification to be triggered.

attributeIdentifier attribute (UInt8)

0 = Reading value

1 to 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 to 255 = Reserved

level attribute (String16)

Contains the preferred schema and extensibility level indication such as "+S1".

limit attribute (UInt32)

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.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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. This attribute SHALL be a fully-qualified absolute URI, not a relative reference.

status attribute (UInt8)

0 = Default status

1 = Subscription canceled, no additional information

2 = Subscription canceled, resource moved

Copyright © 2018 IEEE. All rights reserved.

3 = Subscription canceled, resource definition changed (e.g., a new version of IEEE Std 2030.5)

4 = Subscription canceled, resource deleted

All other values reserved.

subscriptionURI attribute (anyURI)
The Subscription from which this Notification was triggered. This attribute SHALL be a fully-qualified absolute URI, not a relative reference.

NotificationList object (List)
A List element to hold Notification objects.

B.2.7 Response package

Contains definitions of objects enabling responses to be sent back to suppliers and providers.

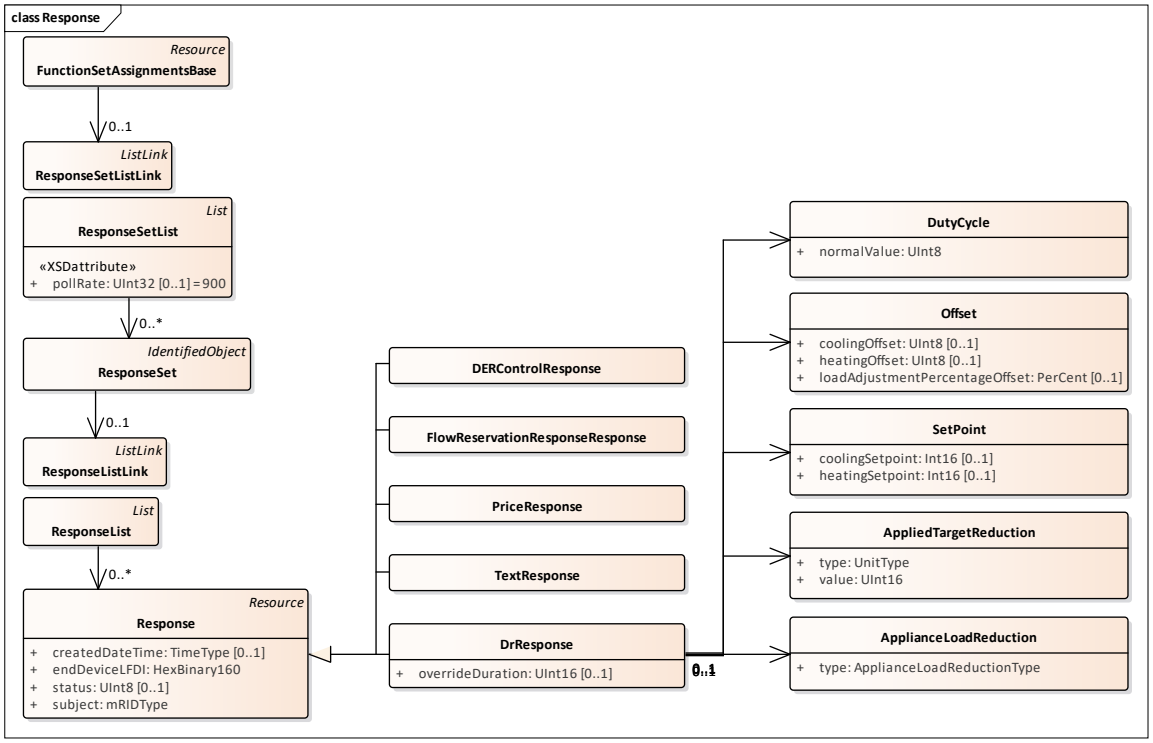


Figure B.13—Response

DERControlResponse object (Response)
A response to a DERControl.

FlowReservationResponseResponse object (Response)
A response to a FlowReservationResponse.

AppliedTargetReduction object ()
Specifies the value of the TargetReduction applied by the device.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Response object (Resource)
The Response object is the generic response data repository which is extended for specific function sets.

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 long-form device identifier (LFDI) of the device providing the response.

status attribute (UInt8) [0..1]
The status field contains the acknowledgement or status. Each event type (DRLC, DER, Price, or Text) can return different status information (e.g., an Acknowledge will be returned for a Price event where a Demand Response Load Control (DRLC) event can return Event Received, Event Started, and Event Completed). The status field value definitions are defined in Table 27.

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.

Copyright © 2018 IEEE. All rights reserved.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

TextResponse object (Response)

A response to a text message.

B.2.8 Time package

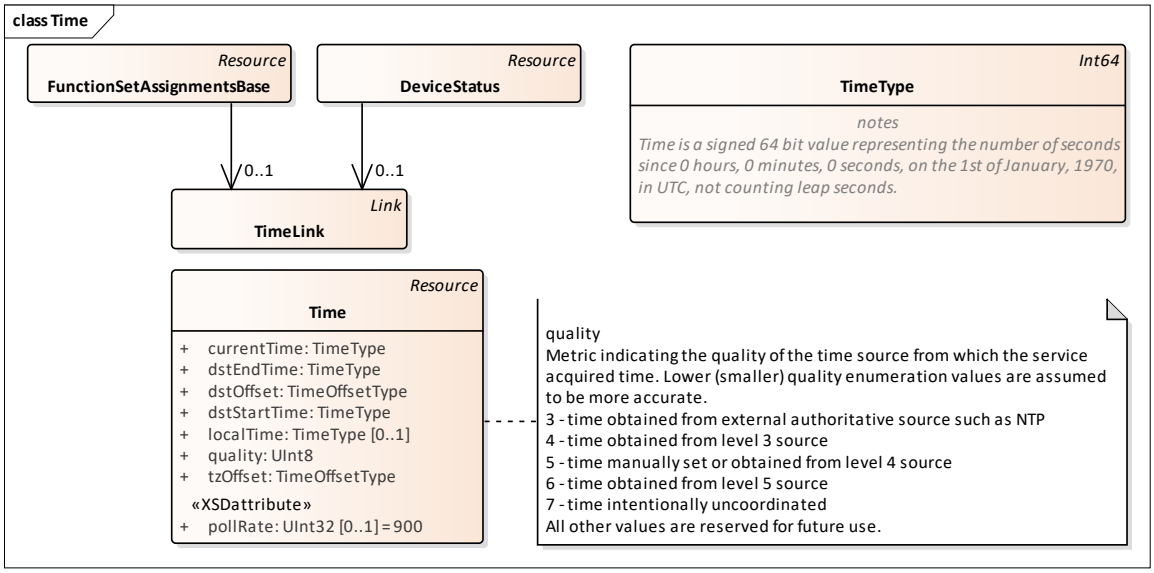


Figure B.14—Time

Time object (Resource)

Contains the representation of time, constantly updated.

currentTime attribute (TimeType)

The current time, in the format defined by TimeType.

dstEndTime attribute (TimeType)

Time at which daylight savings ends (dstOffset no longer applied). Result of dstEndRule calculation.

dstOffset attribute (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 attribute (TimeType)

Time at which daylight savings begins (apply dstOffset). Result of dstStartRule calculation.

localTime attribute (TimeType) [0..1]

Local time: localTime = currentTime + tzOffset (+ dstOffset when in effect).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

quality attribute (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.

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 (e.g., EST = GMT −5 which is −18 000).

B.2.9 DeviceInformation package

Contains general information about devices.

Copyright © 2018 IEEE. All rights reserved.

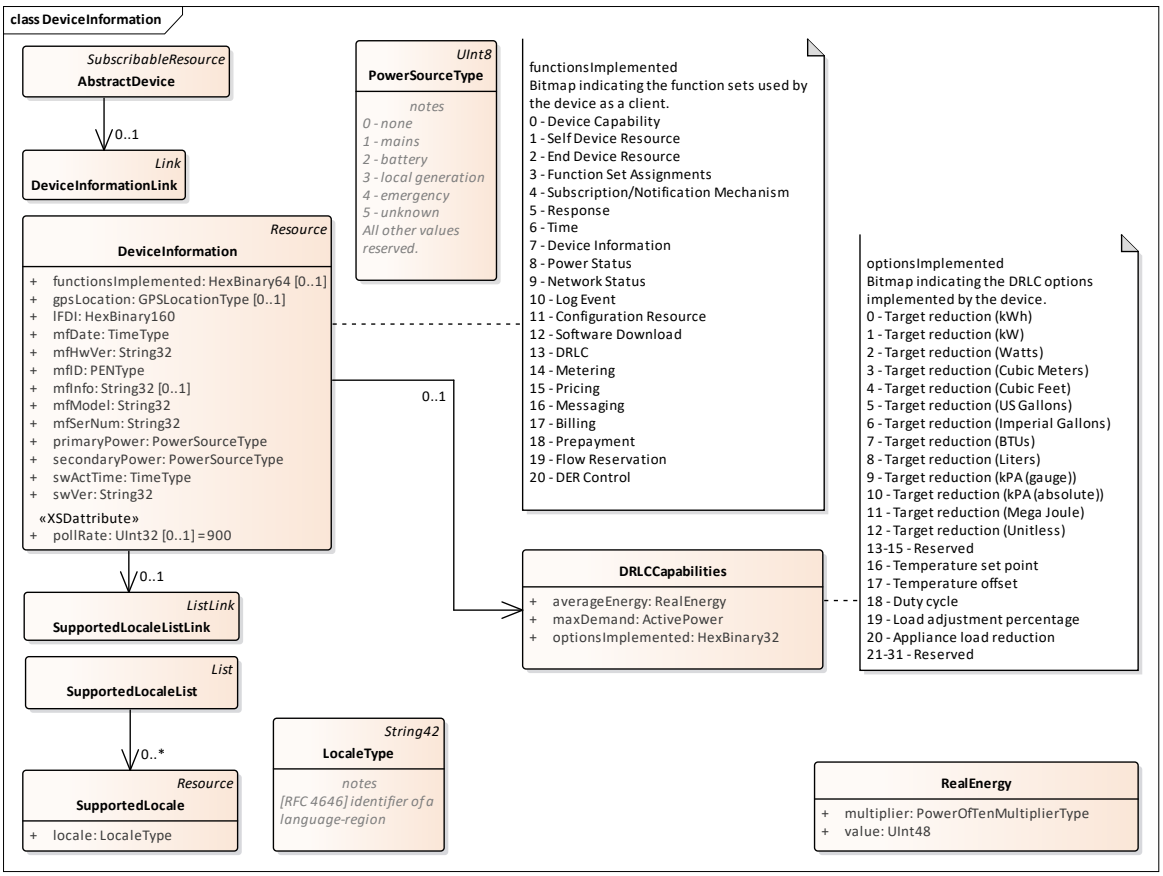


Figure B.15—DeviceInformation

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 = SelfDevice resource

2 = EndDevice resource

3 = Function set assignments

4 = Subscription/Notification mechanism

5 = Response

6 = Time

7 = Device information

8 = Power status

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

9 = Network status

10 = Log event

11 = Configuration resource

12 = Software download

13 = DRLC

14 = Metering

15 = Pricing

16 = Messaging

17 = Billing

18 = Prepayment

19 = Flow reservation

20 = DER control

gpsLocation attribute (GPSLocationType) [0..1]
GPS location of this device.

lFDI attribute (HexBinary160)
Long form device identifier. See Clause 6 (security) 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.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

primaryPower attribute (PowerSourceType)
Primary source of power.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 (U.S. 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 (megajoule)

12 = Target reduction (unitless)

13 to 15 = Reserved

16 = Temperature set point

17 = Temperature offset

Copyright © 2018 IEEE. All rights reserved.

18 = Duty cycle

19 = Load adjustment percentage

20 = Appliance load reduction

21 to 31 = Reserved

SupportedLocale object (Resource)
Specifies a locale that is supported.

locale attribute (LocaleType)
The code for a locale that is supported.

SupportedLocaleList object (List)
A List element to hold SupportedLocale objects.

B.2.10 PowerStatus package

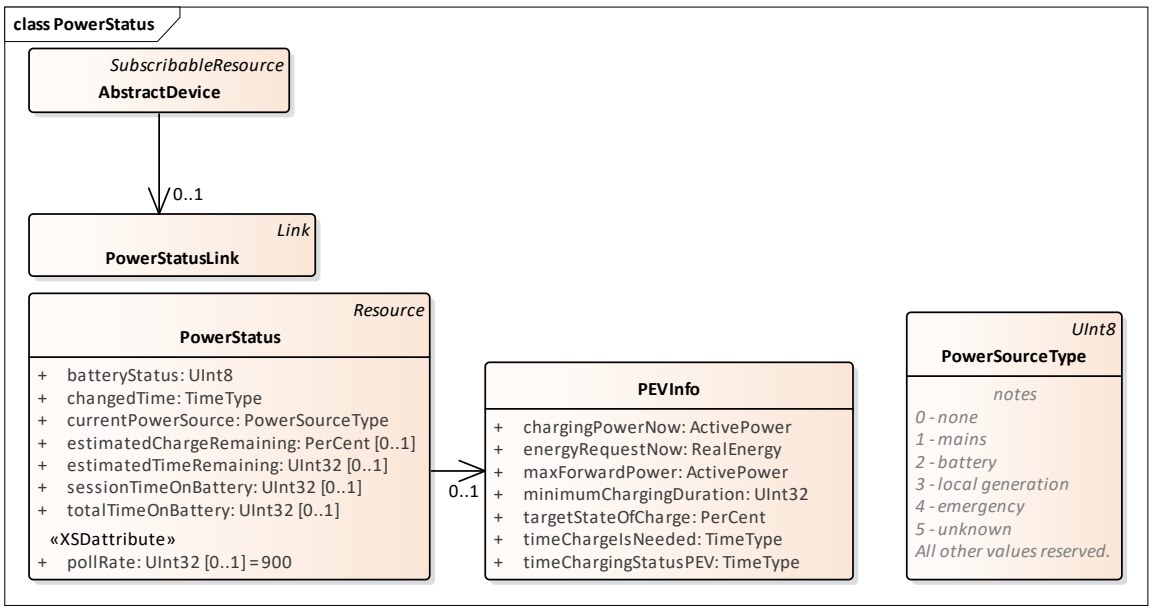


Figure B.16—PowerStatus

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)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 (undercurrent load).

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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

Copyright © 2018 IEEE. All rights reserved.

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 the 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 Time Charge Is Needed (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.

timeChargeIsNeeded 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

B.2.11 NetworkStatus package

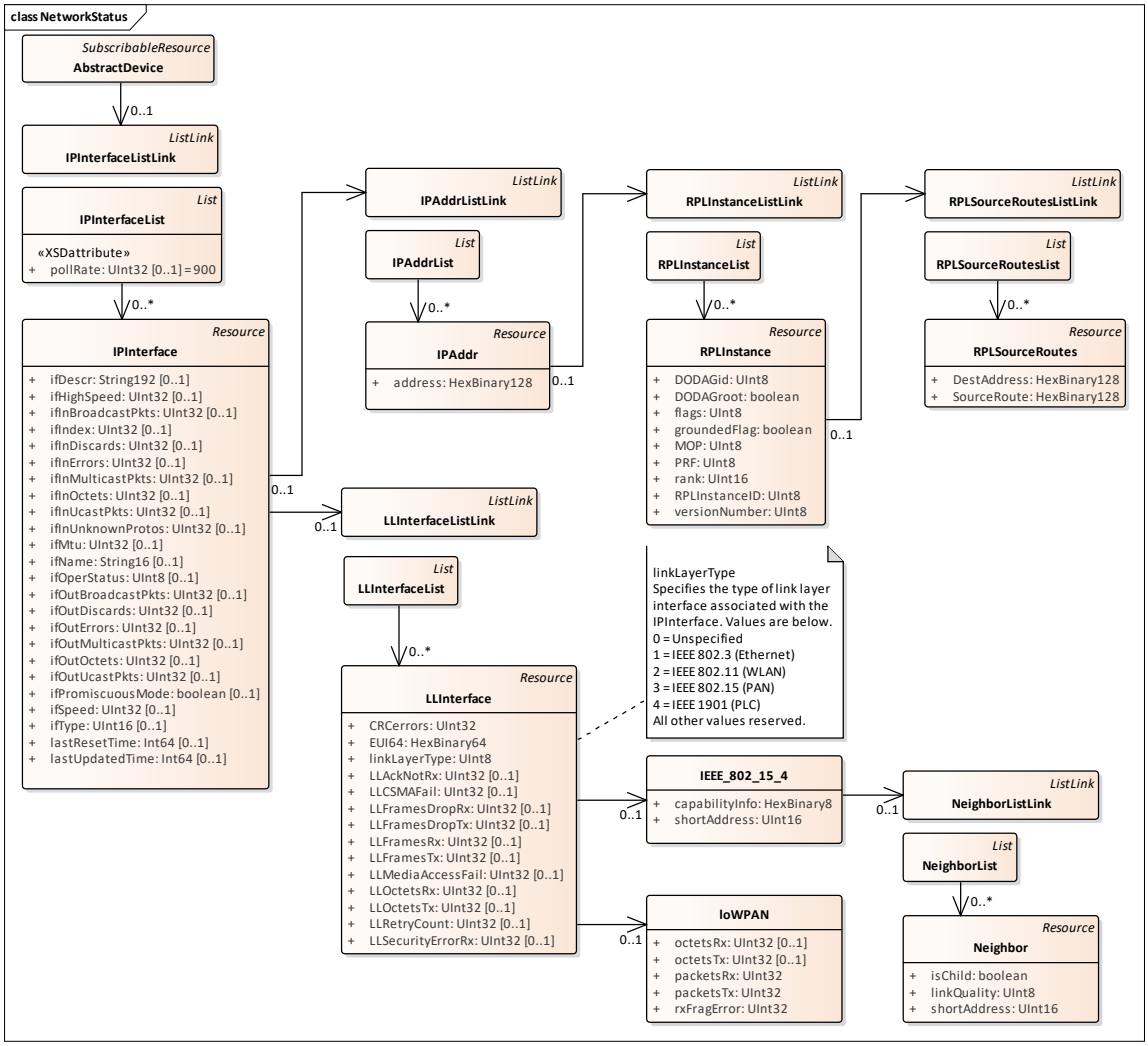


Figure B.17—NetworkStatus

IEEE_802_15_4 object ()

Contains 802.15.4 link layer specific attributes.

capabilityInfo attribute (HexBinary8)

As defined by IEEE Std 802.15.4.

shortAddress attribute (UInt16)

As defined by IEEE Std 802.15.4.

IPAddr object (Resource)

An Internet Protocol (IP) address object.

address attribute (HexBinary128)

An IP address value.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 IETF RFC 2863.

ifHighSpeed attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

ifInBroadcastPkts attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

ifIndex attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

ifInDiscards attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Input Datagrams Discarded.

ifInErrors attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

ifInMulticastPkts attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Multicast Datagrams Received.

ifInOctets attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Bytes Received.

ifInUcastPkts attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Datagrams Received.

ifInUnknownProtos attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Datagrams with Unknown Protocol Received.

ifMtu attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

ifName attribute (String16) [0..1]
Use rules from IETF RFC 2863.

ifOperStatus attribute (UInt8) [0..1]
Use rules and assignments from IETF RFC 2863.

ifOutBroadcastPkts attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Broadcast Datagrams Sent.

ifOutDiscards attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Output Datagrams Discarded.

ifOutErrors attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

ifOutMulticastPkts attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Multicast Datagrams Sent.

ifOutOctets attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Bytes Sent.

ifOutUcastPkts attribute (UInt32) [0..1]
Use rules from IETF RFC 2863. Can be thought of as Datagrams Sent.

ifPromiscuousMode attribute (boolean) [0..1]
Use rules from IETF RFC 2863.

ifSpeed attribute (UInt32) [0..1]
Use rules from IETF RFC 2863.

ifType attribute (UInt16) [0..1]
Use rules and assignments from IETF RFC 2863.

lastResetTime attribute (Int64) [0..1]
Similar to ifLastChange in IETF RFC 2863.

lastUpdatedTime attribute (Int64) [0..1]
The date/time of the reported status.

IPInterfaceList object (List)
List of IPInterface instances.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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 IETF 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.

Copyright © 2018 IEEE. All rights 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.

loWPAN 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 IEEE Std 802.15.4.

shortAddress attribute (UInt16)
As defined by IEEE Std 802.15.4.

NeighborList object (List)
List of 802.15.4 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 IETF RFC 6550.

DODAGroot attribute (boolean)
See IETF RFC 6550.

flags attribute (UInt8)
See IETF RFC 6550.

groundedFlag attribute (boolean)
See IETF RFC 6550.

MOP attribute (UInt8)
See IETF RFC 6550.

PRF attribute (UInt8)
See IETF RFC 6550.

rank attribute (UInt16)
See IETF RFC 6550.

RPLInstanceID attribute (UInt8)
See IETF RFC 6550.

versionNumber attribute (UInt8)
See IETF RFC 6550.

RPLInstanceList object (List)
List of RPLInstances associated with the IPInterface.

RPLSourceRoutes object (Resource)
A RPL source routes object.

DestAddress attribute (HexBinary128)
See IETF RFC 6554.

Copyright © 2018 IEEE. All rights reserved.

SourceRoute attribute (HexBinary128)
See IETF RFC 6554.

RPLSourceRoutesList object (List)
List of RPL source routes if the hosting device is the DODAGroot.

B.2.12 LogEvents package

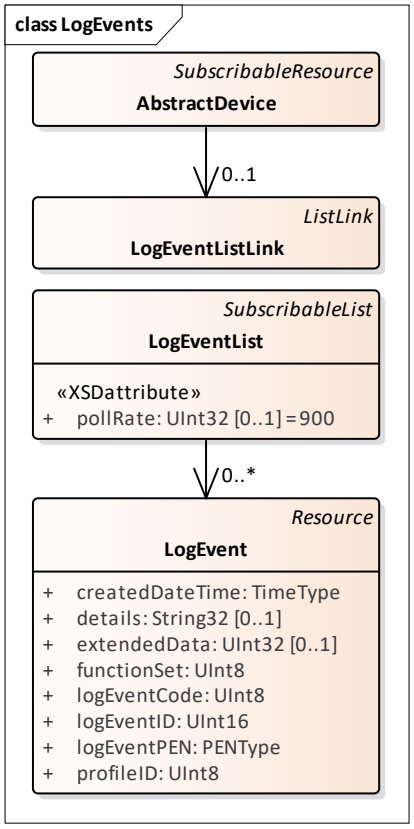


Figure B.18—LogEvents

LogEvent object (Resource)
A time-stamped instance of a significant event detected by the device.

createdDateTime attribute (TimeType)
The date and time that the event occurred.

details attribute (String32) [0..1]
Human-readable text that MAY be used to transmit additional details about the event. A host MAY remove this field when received.

extendedData attribute (UInt32) [0..1]
May be used to transmit additional details about the event.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

functionSet attribute (UInt8)

If the profileID indicates this is IEEE 2030.5, the functionSet is defined by IEEE Std 2030.5 and SHALL be one of the values from the list below (IEEE 2030.5 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

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 IEEE 2030.5, the logEventCode is defined by IEEE Std 2030.5 within one of the function sets of IEEE Std 2030.5. 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

logEventPEN attribute (PENType)

The Private Enterprise Number (PEN) of the entity that defined the profileID, functionSet, and logEventCode of the logEvent. IEEE Std 2030.5-assigned logEventCodes SHALL use the IEEE 2030.5 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

2 = IEEE 2030.5

3 = Home automation

4 = Building automation

All other values are reserved.

LogEventList object (SubscribableList)

A List element to hold LogEvent objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

Copyright © 2018 IEEE. All rights reserved.

B.2.13 Configuration package

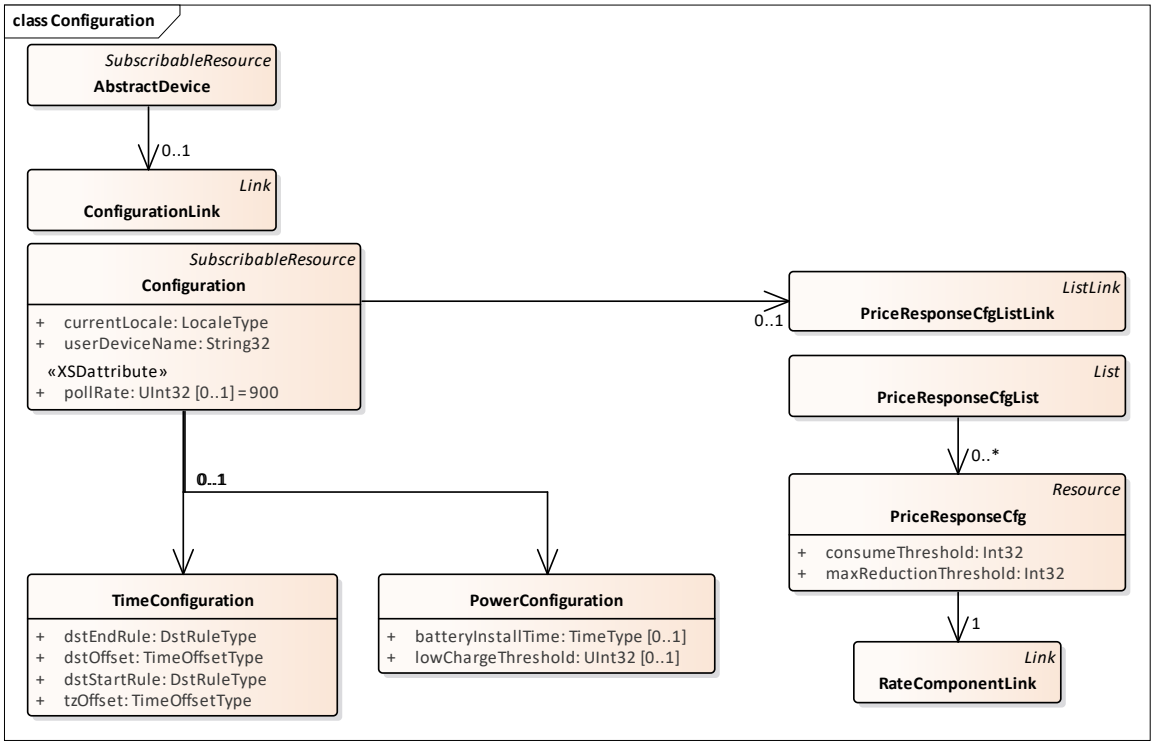


Figure B.19—Configuration

Configuration object (SubscribableResource)

This resource contains various settings to control the operation of the device.

currentLocale attribute (LocaleType)

IETF RFC 4646 identifier of the language-region currently in use.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

userDeviceName attribute (String32)

User assigned, convenience name used for network browsing displays, etc. For 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 PS_LOW_BATTERY is raised.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Copyright © 2018 IEEE. All rights reserved.

B.2.14 SoftwareDownload package

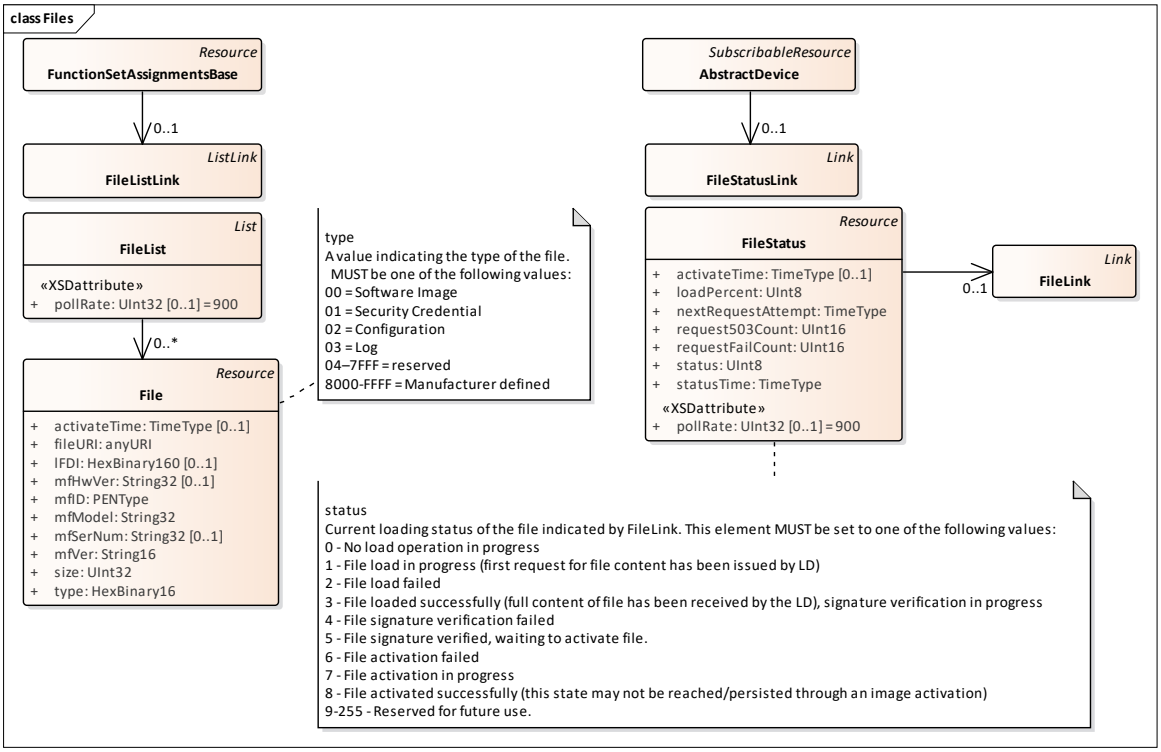


Figure B.20—Files

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 binary large object (BLOB) that is actually loaded by the LD.

lFDI attribute (HexBinary160) [0..1]

This element MUST be set to the LFDI of the device for which this file in 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).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 to 7FFF = reserved

8000 to FFFF = Manufacturer defined

FileList object (List)
A List element to hold File objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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 fails. 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 to 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.

B.2.15 DRLC package

Contains definitions for Demand Response Load Control functionality.

Copyright © 2018 IEEE. All rights reserved.

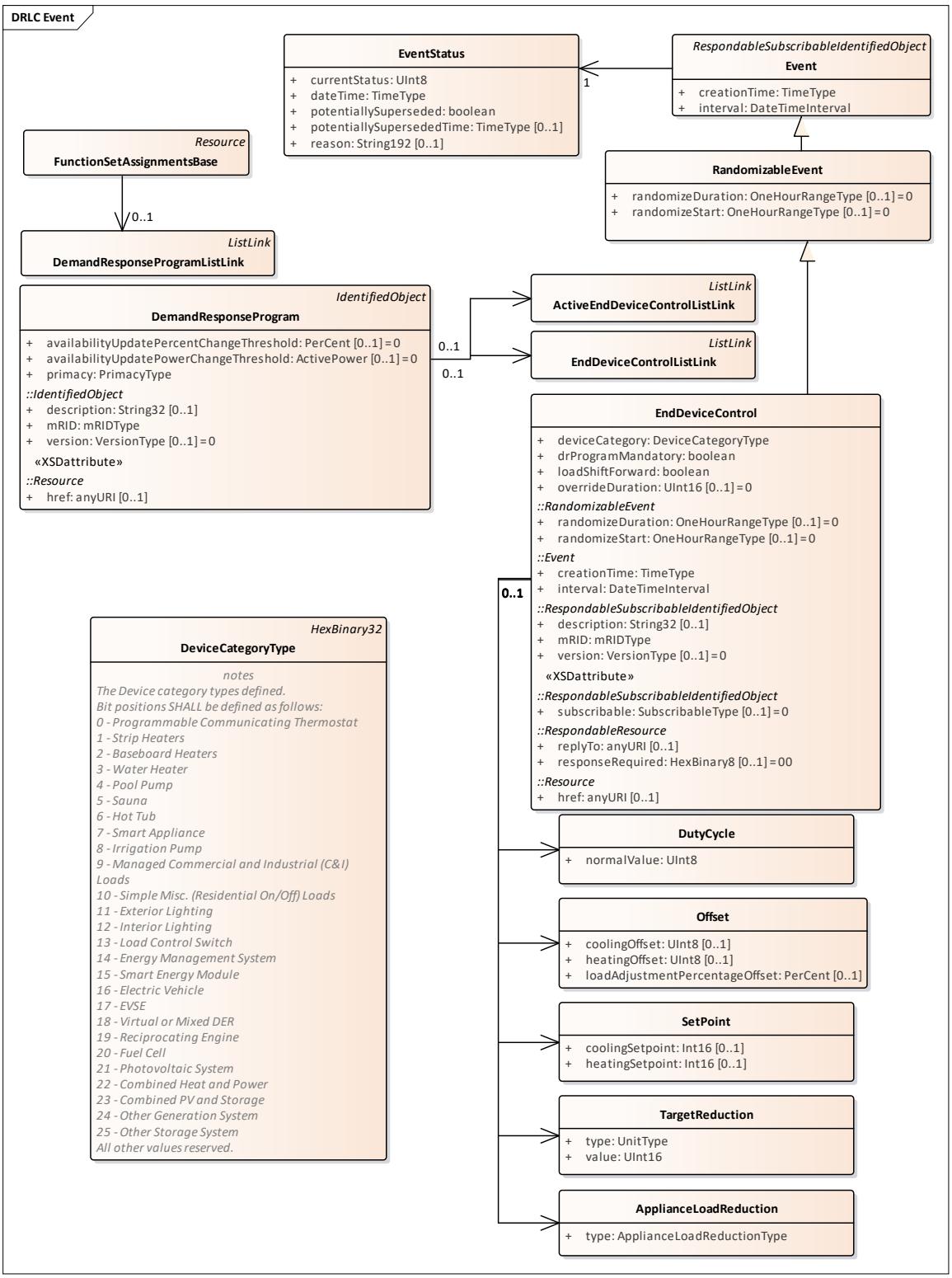


Figure B.21—DRLC event

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

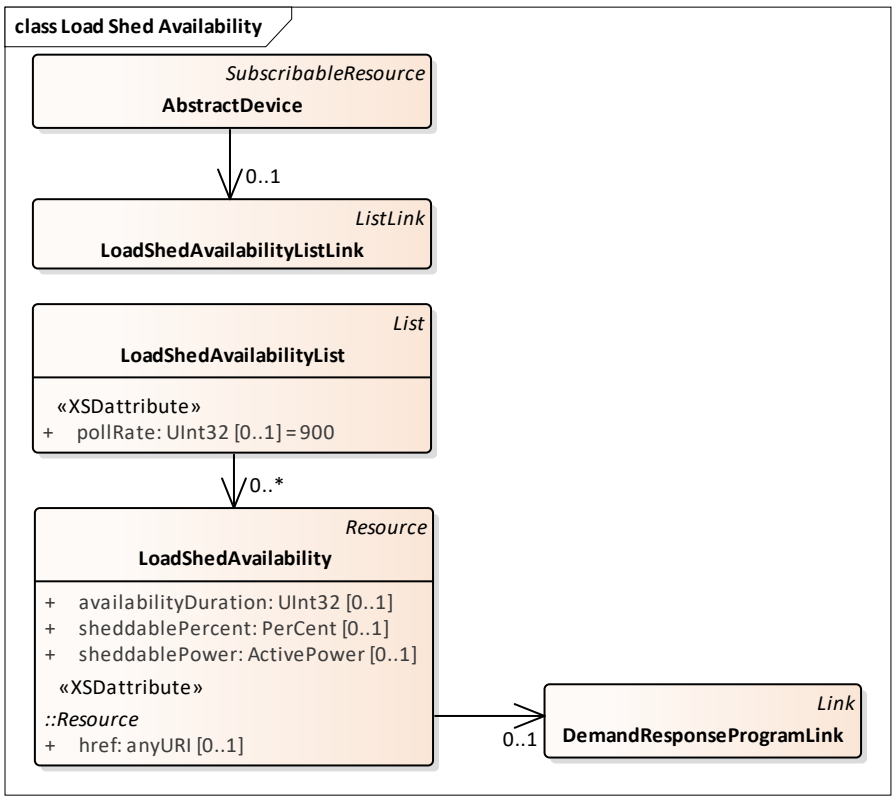


Figure B.22—LoadShedAvailability

LoadShedAvailabilityList Object (List)

A List element to hold LoadShedAvailability objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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 provided. If present and greater than zero, then clients SHALL provide their LoadShedAvailability if it has not previously been provided, and thereafter if the difference between the previously provided value and the current value of LoadShedAvailability sheddablePercent is greater than availabilityUpdatePercentChangeThreshold.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 provided. If present and greater than zero, then clients SHALL provide their LoadShedAvailability if it has not previously been provided, and thereafter if the difference between the previously provided 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.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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 deviceCategory.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Association object ()

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 end device control (EDC) is being targeted at multiple devices or to a device that controls multiple devices (e.g., energy management system [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 °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 °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 (PerCent) [0..1]

The value change requested for the load adjustment percentage. 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:

$$\frac{\text{Cooling/Heating Temperature Set Point}}{100} = \text{temperature in °C where } -273.15\ °C \leq \text{temperature} \leq 327.67\ °C,$$

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.

Copyright © 2018 IEEE. All rights reserved.

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 °C /100 (hundredths of a degree Celsius).

heatingSetpoint attribute (Int16) [0..1]
This attribute represents the heating temperature set point in °C/100 (hundredths of a degree Celsius).

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.

B.2.16 Metering package

Contains definitions related to measurements of energy at usage points.

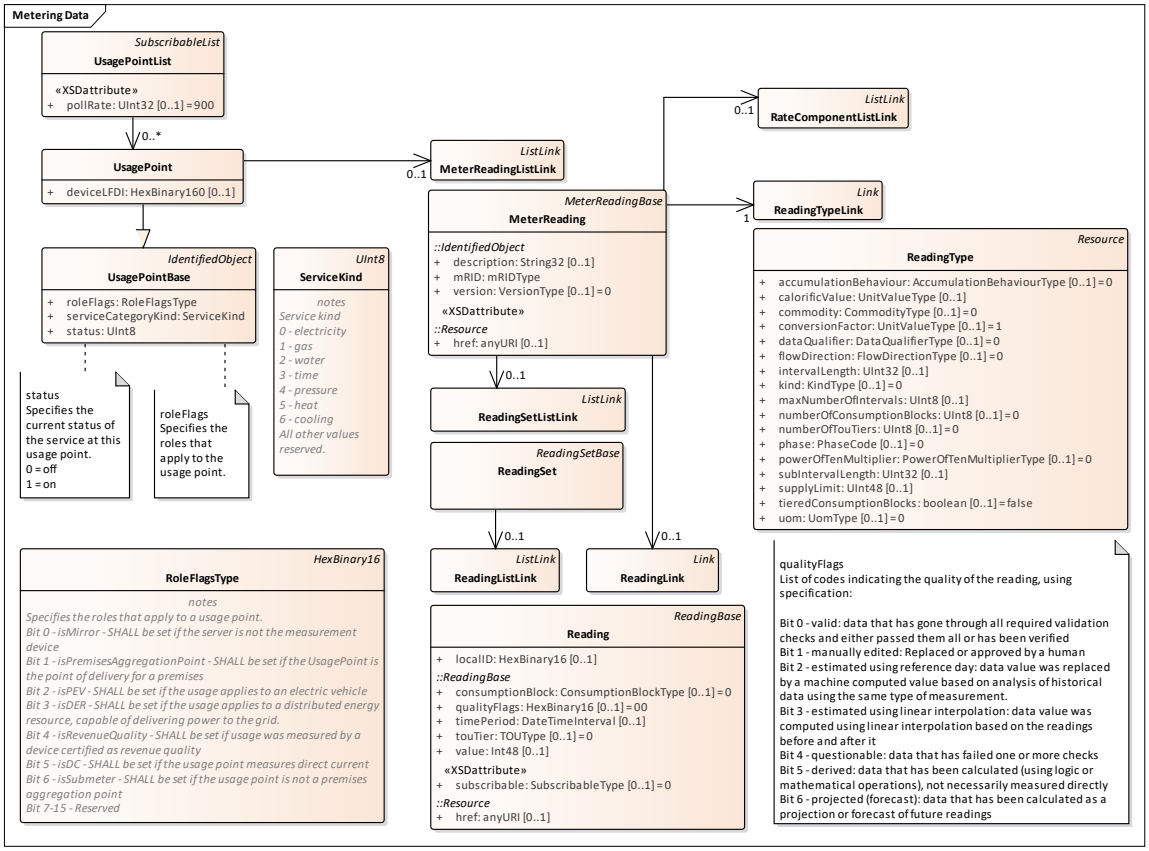


Figure B.23—Metering data

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

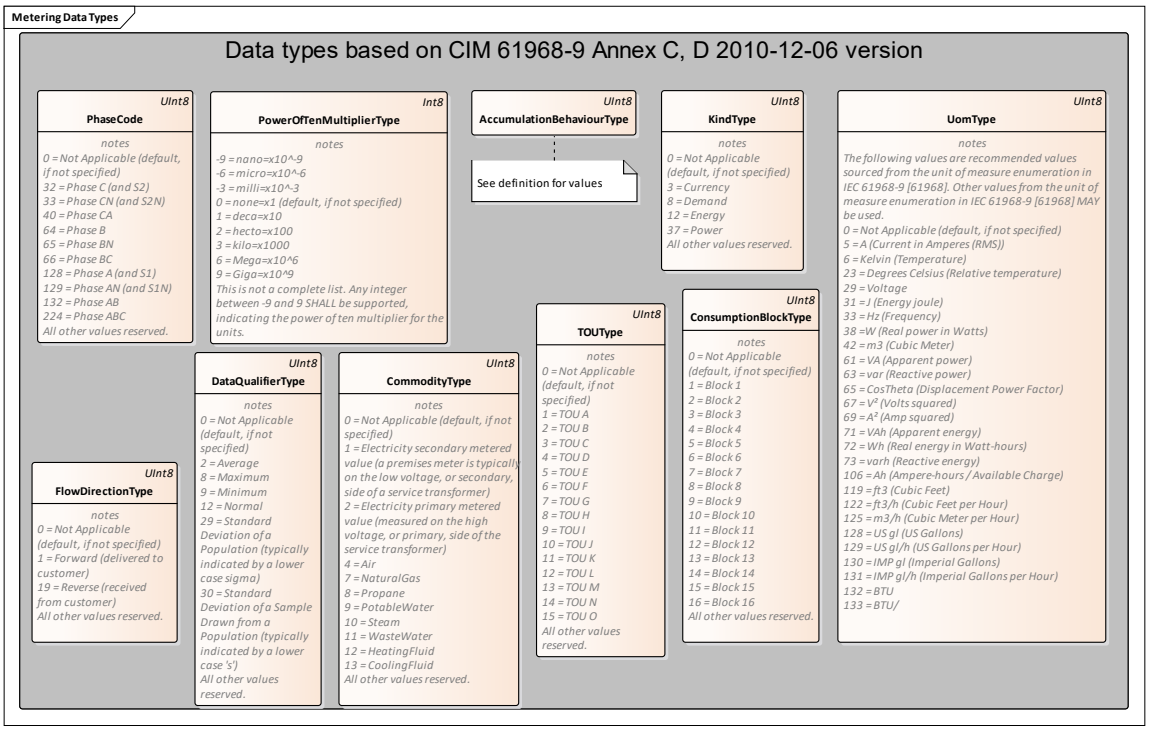


Figure B.24—Metering data types

MeterReading object (MeterReadingBase)
Set of values obtained from the meter.

MeterReadingList object (SubscribableList)
A List element to hold MeterReading objects.

Reading object (ReadingBase)
Specific value measured by a meter or other asset.

localID attribute (HexBinary16) [0..1]
The local identifier for this reading within the reading set. localIDs are assigned in order of creation time. For interval data, this value SHALL increase with each interval time, and for block/tier readings, localID SHALL not be specified.

subscribable attribute (SubscribableType) [0..1] «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 (SubscribableList)
A List element to hold Reading objects.

ReadingSet object (ReadingSetBase)
A set of Readings of the ReadingType indicated by the parent MeterReading.

ReadingSetList object (SubscribableList)
A List element to hold ReadingSet objects.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

ReadingType object (Resource)

Type of data conveyed by a specific Reading. See IEC 61968 Part 9 Annex C for full definitions of these values.

accumulationBehaviour attribute (AccumulationBehaviourType) [0..1]

The accumulation behaviour indicates how the value is represented to accumulate over time.

calorificValue attribute (UnitValueType) [0..1]

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.

commodity attribute (CommodityType) [0..1]

Indicates the commodity applicable to this ReadingType.

conversionFactor attribute (UnitValueType) [0..1]

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.

dataQualifier attribute (DataQualifierType) [0..1]

The data type can be used to describe a salient attribute of the data. Possible values are average, absolute, etc.

flowDirection attribute (FlowDirectionType) [0..1]

Anything involving current might have a flow direction. Possible values include forward and reverse.

intervalLength attribute (UInt32) [0..1]

Default interval length specified in seconds.

kind attribute (KindType) [0..1]

Compound class that contains kindCategory and kindIndex.

maxNumberOfIntervals attribute (UInt8) [0..1]

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.

numberOfConsumptionBlocks attribute (UInt8) [0..1]

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.

numberOfTouTiers attribute (UInt8) [0..1]

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 critical peak pricing (CPP) tiers that may be used while this ReadingType is in effect. Servers SHALL specify a value between 0 and 255 (inclusive) for numberOfTouTiers (servers providing flat rate pricing SHOULD set numberOfTouTiers to 0, as in practice there is no difference between having no tiers and having one tier).

phase attribute (PhaseCode) [0..1]

Contains phase information associated with the type.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

powerOfTenMultiplier attribute (PowerOfTenMultiplierType) [0..1]
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

uom attribute (UomType) [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).

deviceLFDI attribute (HexBinary160) [0..1]
The LFDI of the source device. This attribute SHALL be present when mirroring.

UsagePointList object (SubscribableList)
A List element to hold UsagePoint objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

Copyright © 2018 IEEE. All rights reserved.

B.2.16.1 MeteringMirror package

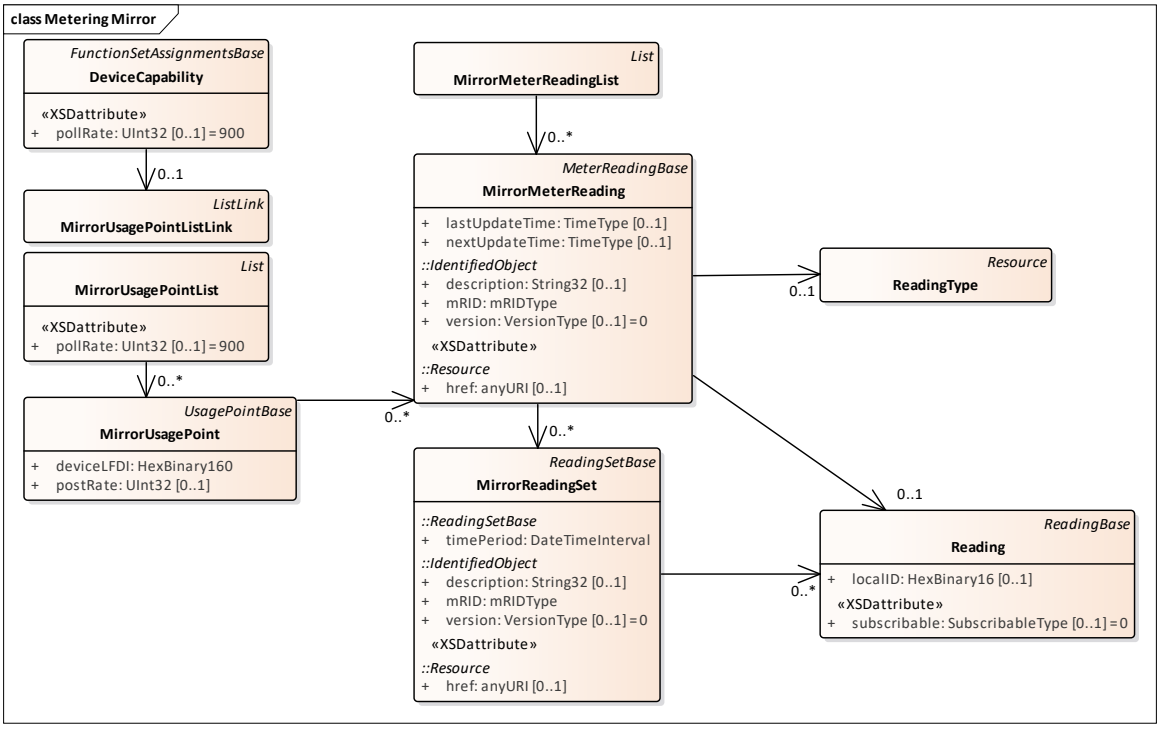


Figure B.25—Metering mirror

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

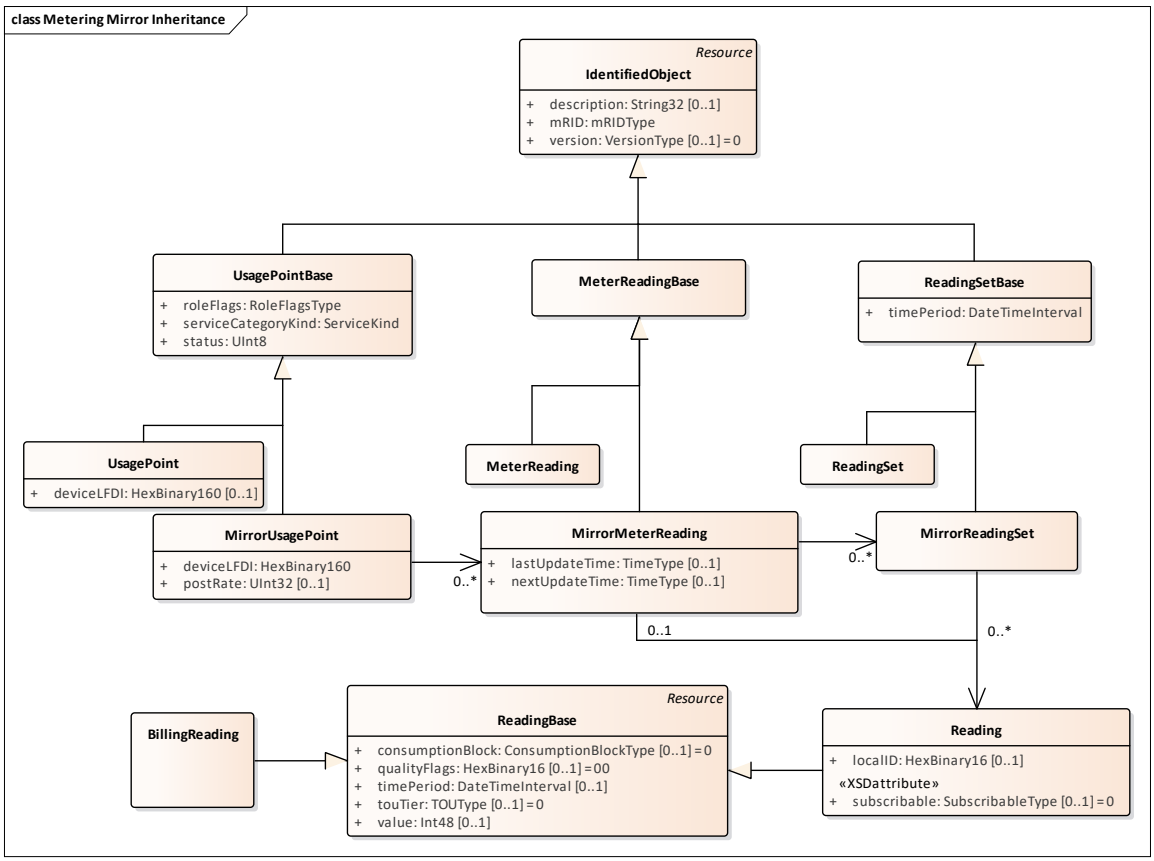


Figure B.26—Metering mirror inheritance

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

postRate attribute (UInt32) [0..1]
POST rate, or how often mirrored data should be POSTed, in seconds. A client MAY indicate a preferred postRate when POSTing MirrorUsagePoint. A server MAY add or modify postRate to indicate its preferred posting rate.

MirrorUsagePointList object (List)
A list of MirrorUsagePoint instances.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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 have 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 have failed one or more checks.

Bit 5—derived: Data that have been calculated (using logic or mathematical operations), not necessarily measured directly.

Bit 6—projected (forecast): Data that have 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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

B.2.17 Pricing package

Contains definitions of information related to price.

Copyright © 2018 IEEE. All rights reserved.

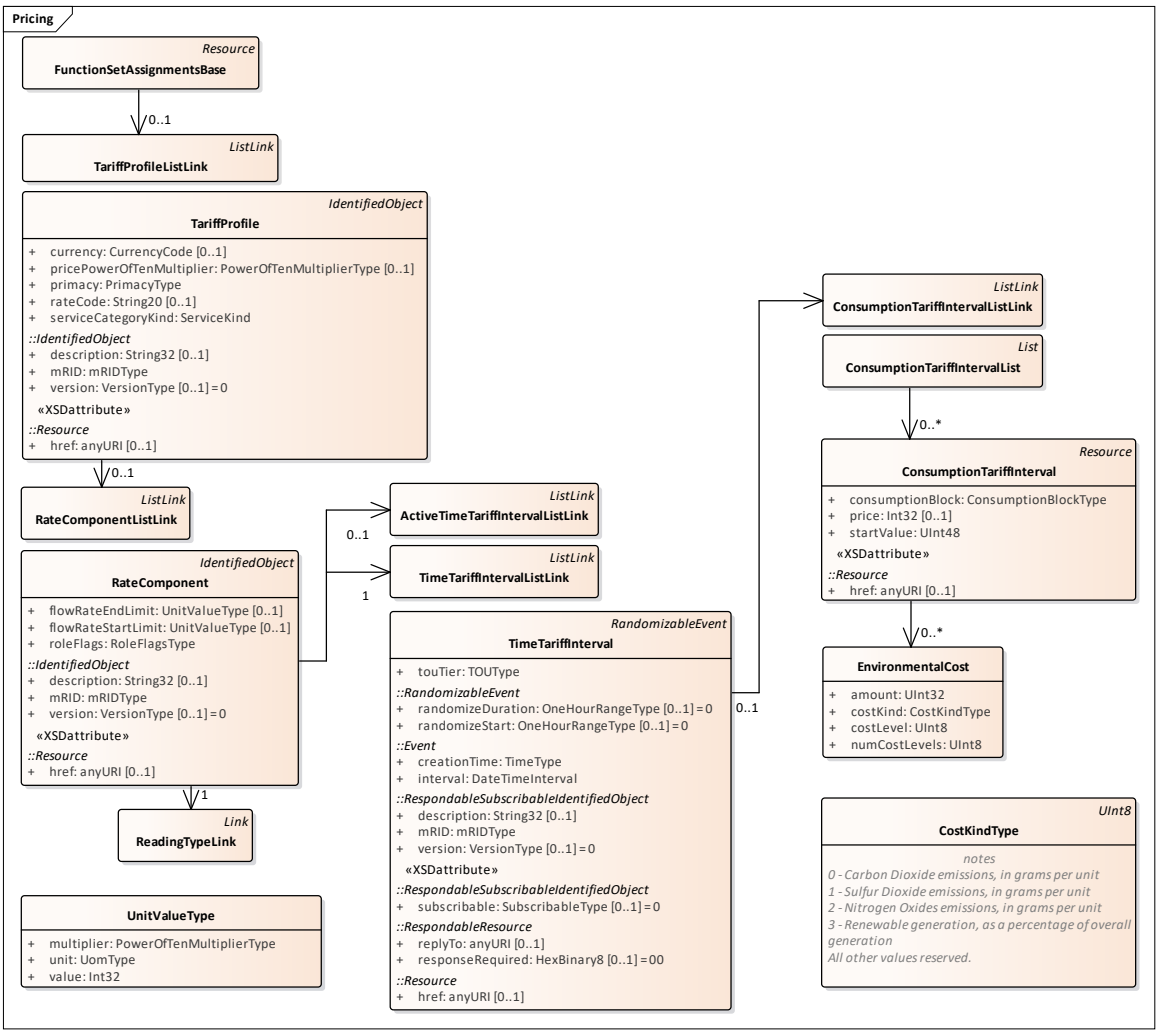


Figure B.27—Pricing

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 TimeTariffInteral 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.

Copyright © 2018 IEEE. All rights reserved.

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 kW of demand for the given rate structure. Above this threshold (from 20 kW to 50 kW), 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 IEEE 2030.5 community can then use as source material for the next version of IEEE Std 2030.5.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

pollRate attribute (Uint32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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.

B.2.18 Messaging package

Contains text message definitions.

Copyright © 2018 IEEE. All rights reserved.

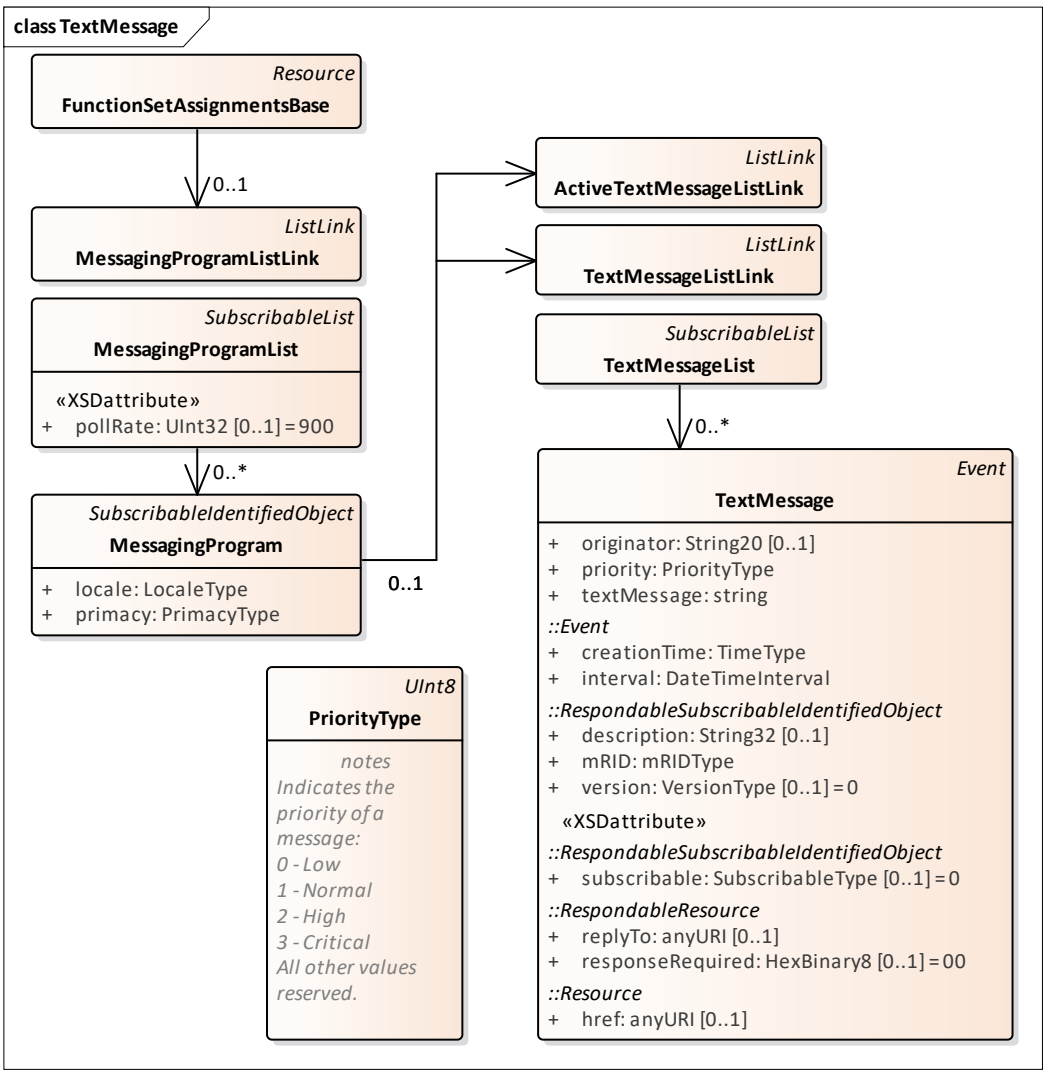


Figure B.28—TextMessage

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.

pollRate attribute (UInt32) [0..1] «XSDattribute»

The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

B.2.19 Billing package

Contains representations of charges and other billing related information.

Copyright © 2018 IEEE. All rights reserved.

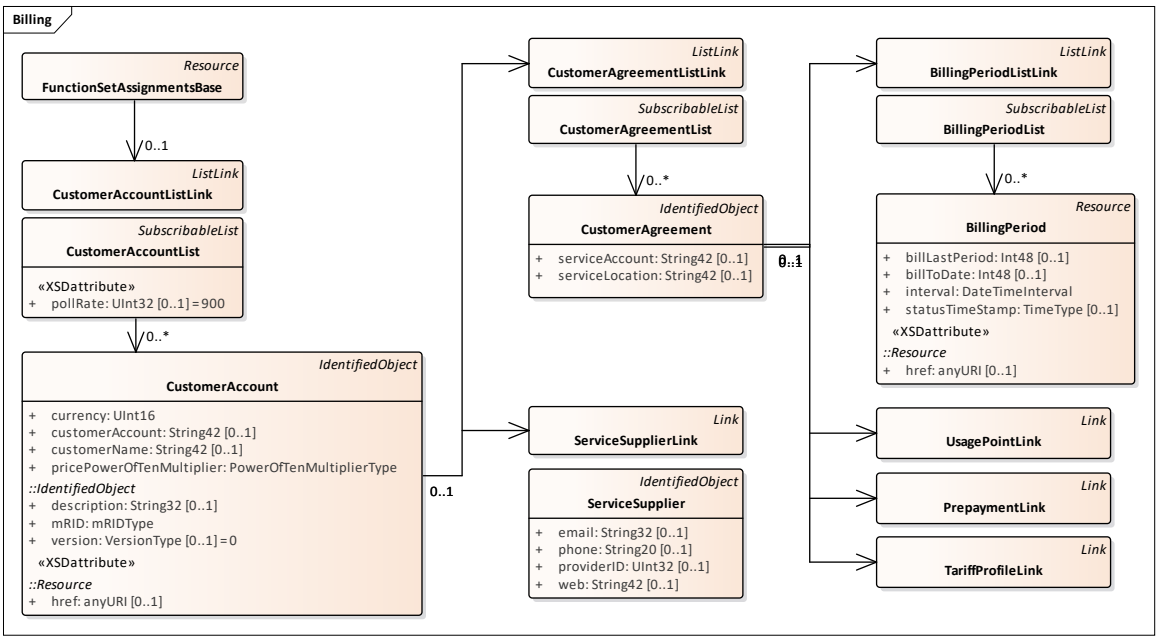


Figure B.29—Billing

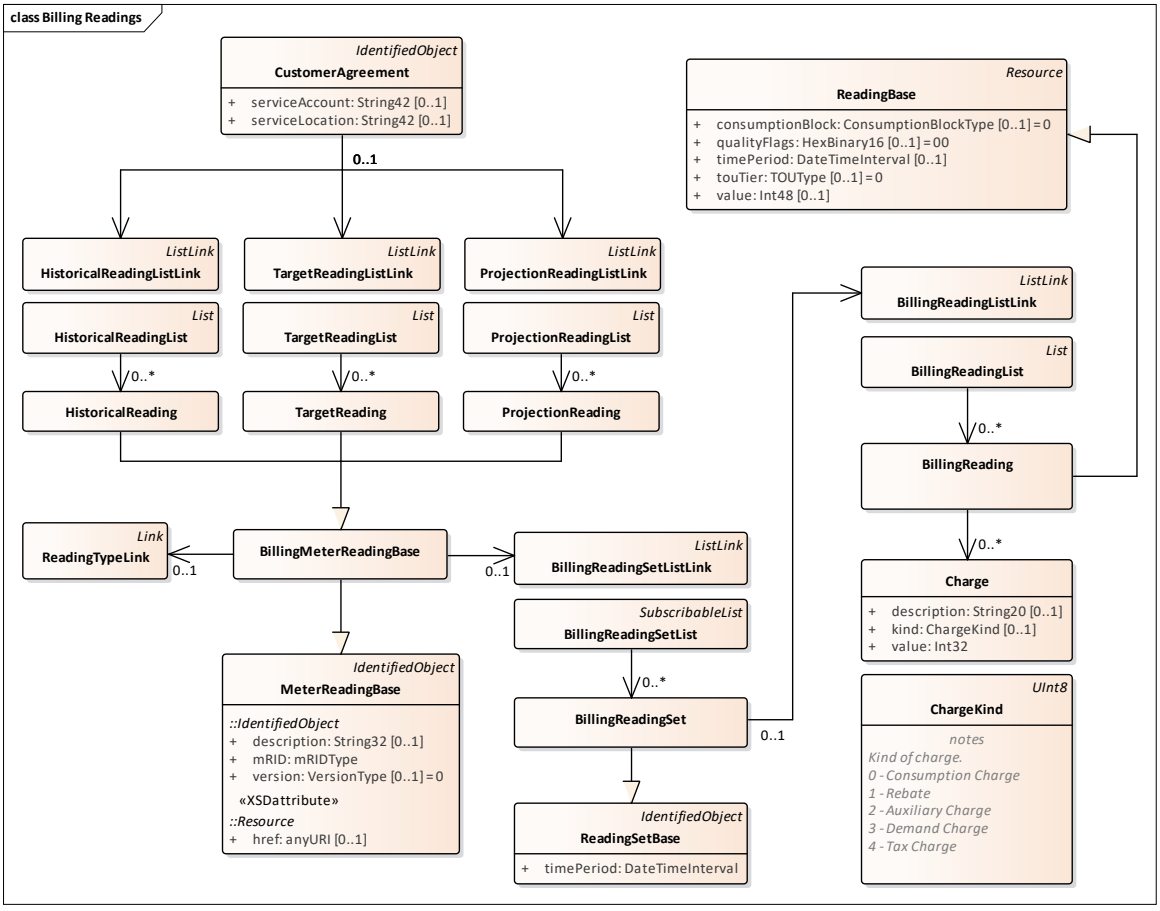


Figure B.30—Billing readings

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

BillingPeriod object (Resource)

BillingPeriod 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 1 July 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 minutes, 10 minutes, 15 minutes, 30 minutes, or 60 minutes. 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 ()

Charge contains 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 is 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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)
Organization 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.

B.2.20 Prepayment package

Contains definitions related to storing and using payments.

Copyright © 2018 IEEE. All rights reserved.

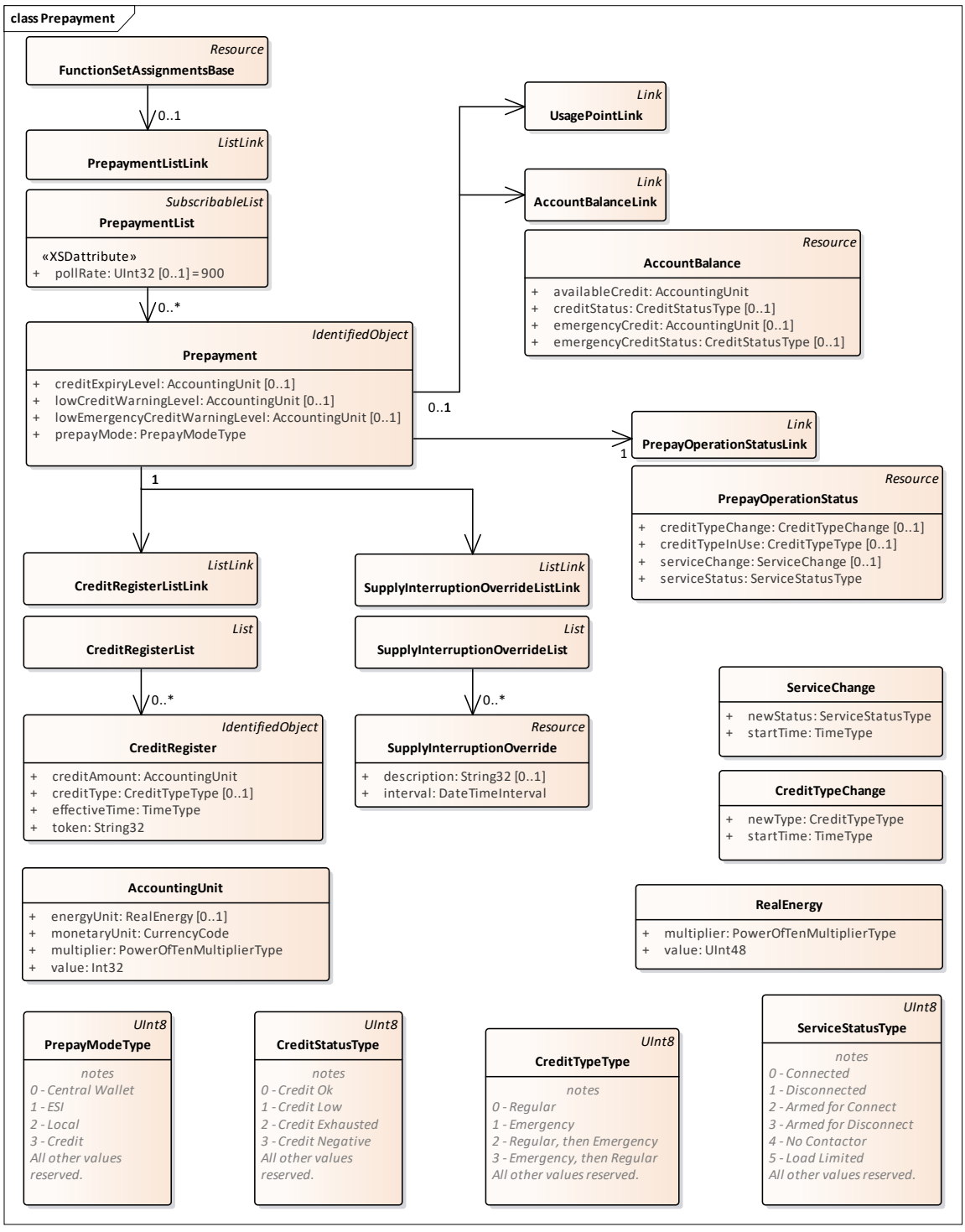


Figure B.31—Prepayment

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

emergencyCredit attribute (AccountingUnit) [0..1]
EmergencyCredit is the amount of credit still available for the given service or commodity prepayment instance. If both availableCredit and emergyCredit are exhausted, then service will typically be disconnected.

emergencyCreditStatus 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

token attribute (String32)
Token is security data that authenticates the legitimacy of the transaction. The details of this token are not defined by IEEE Std 2030.5. 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 10.8.1 (overview).

PrepaymentList object (SubscribableList)
A List element to hold Prepayment objects.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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)
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

Copyright © 2018 IEEE. All rights reserved.

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.

B.2.21 FlowReservation package

Contains flow (charge) reservation model to allow fine-grained control of high-demand loads such as fast-charging batteries.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

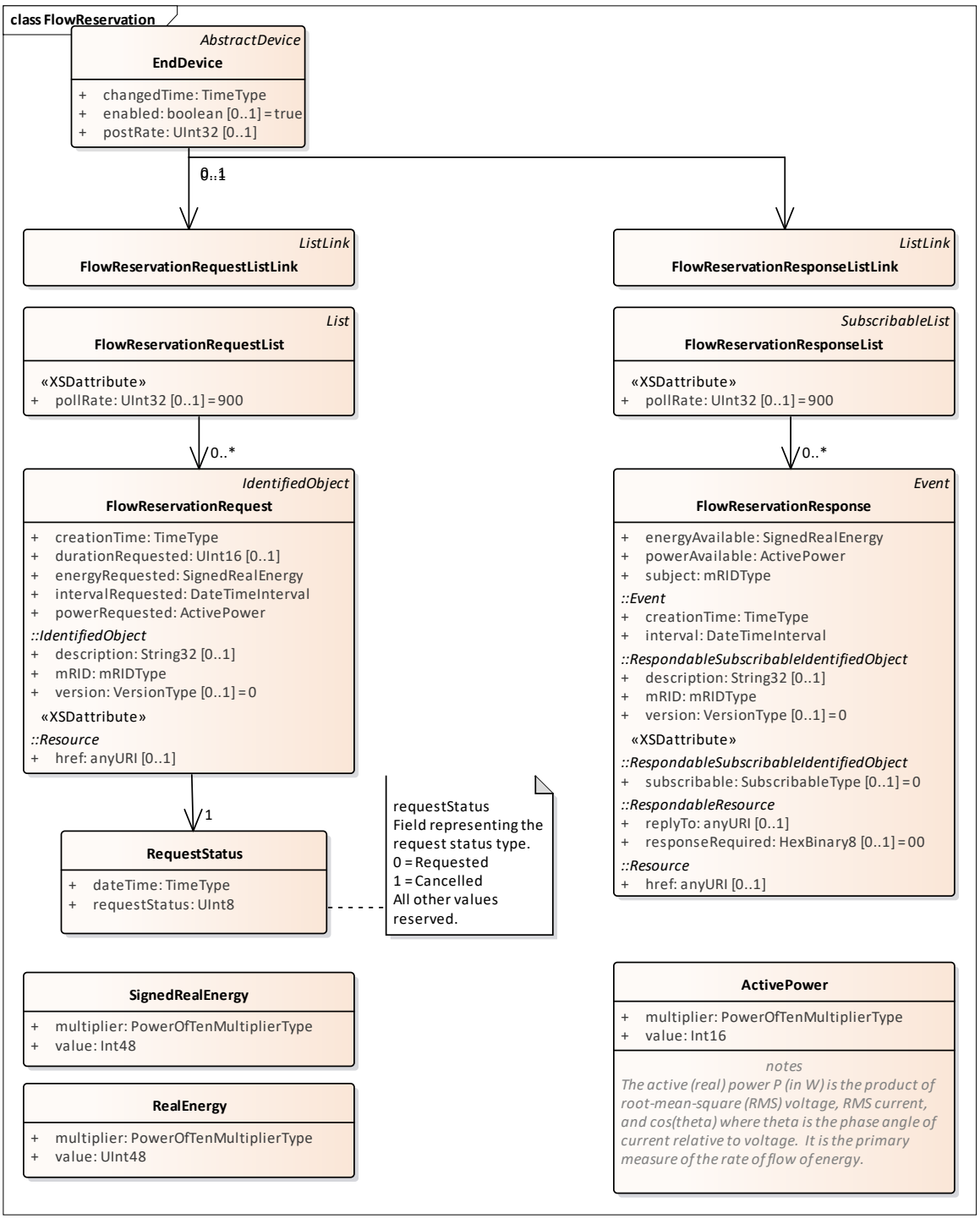


Figure B.32—FlowReservation

RequestStatus object ()

The RequestStatus object is used to indicate the current status of a Flow Reservation Request.

dateTime attribute (TimeType)

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

requestStatus attribute (UInt8)
Field representing the request status type.

0 = Requested

1 = Cancelled

All other values reserved.

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 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.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

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 (SignedRealEnergy)
Indicates the amount of energy available.

Copyright © 2018 IEEE. All rights reserved.

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.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

B.2.22 DER package

Contains definitions related to allowing distributed energy resources to provide energy back to the grid.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

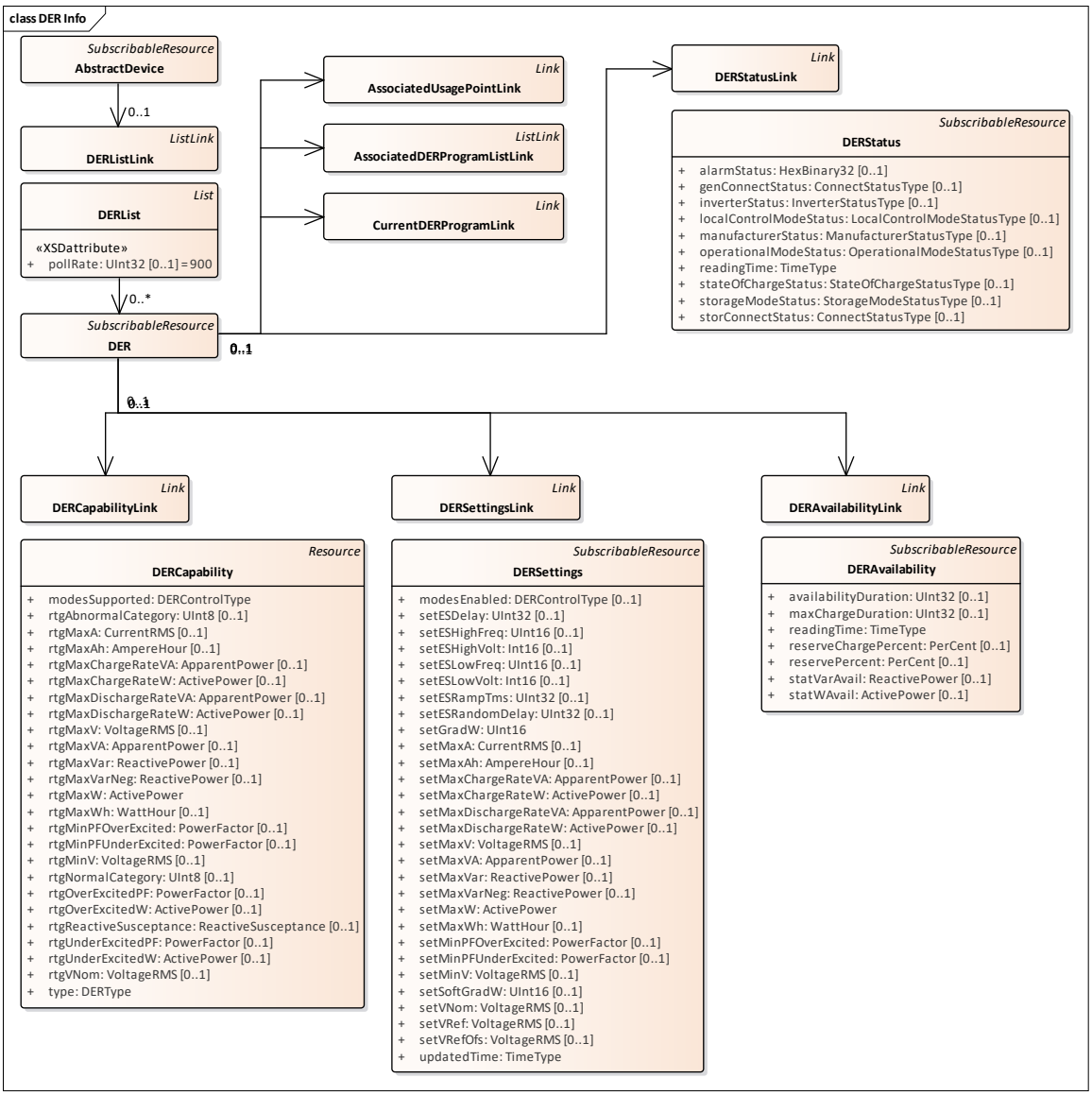


Figure B.33—DER info

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

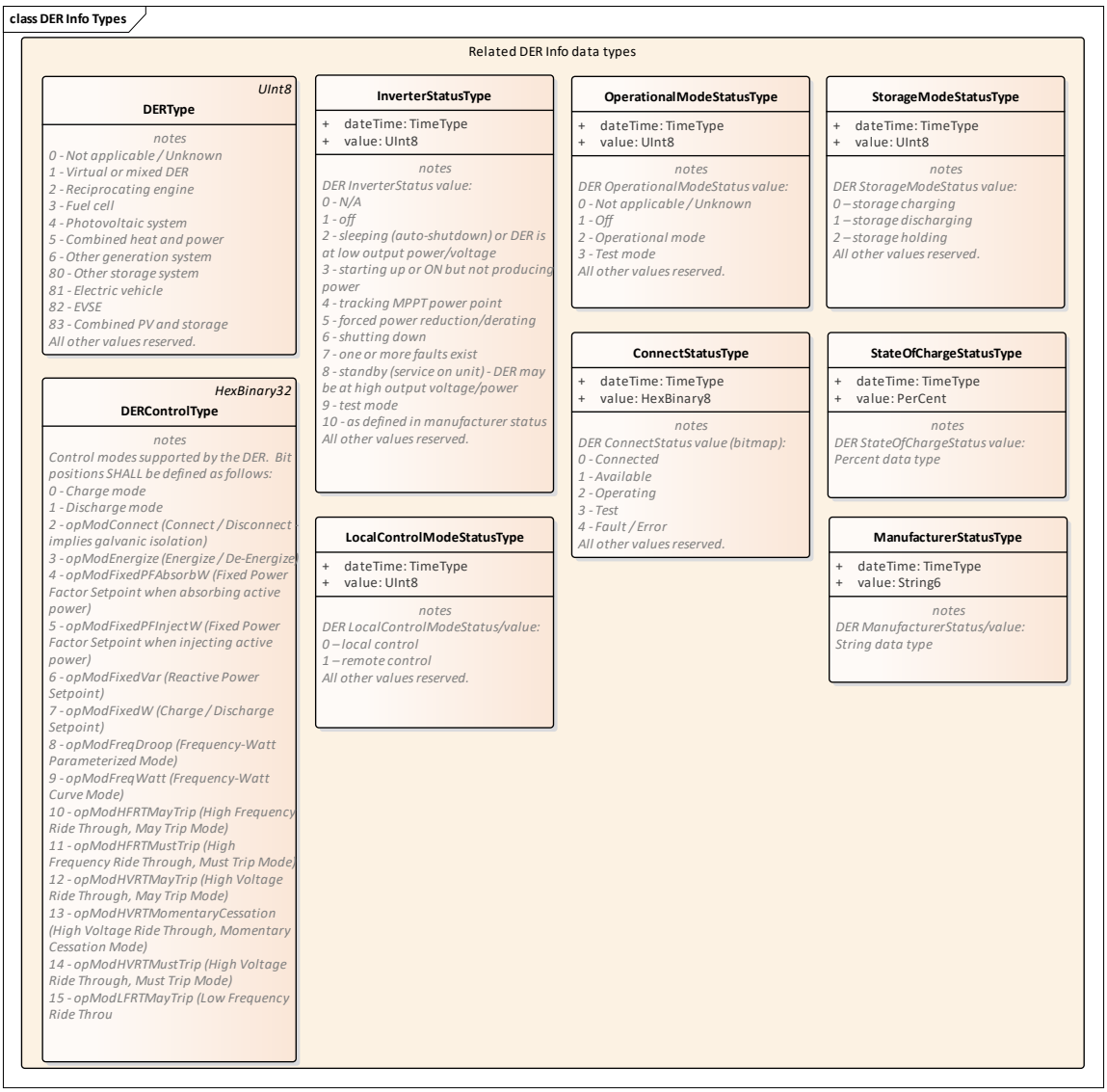


Figure B.34—DER info types

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

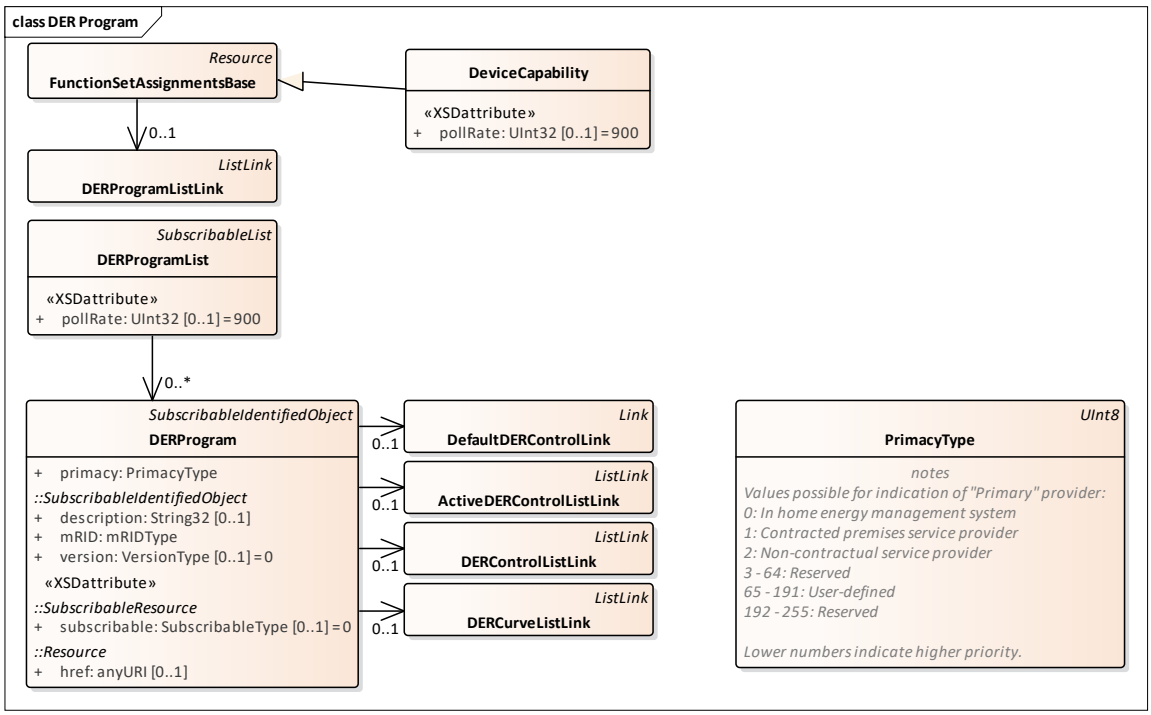


Figure B.35—DER program

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

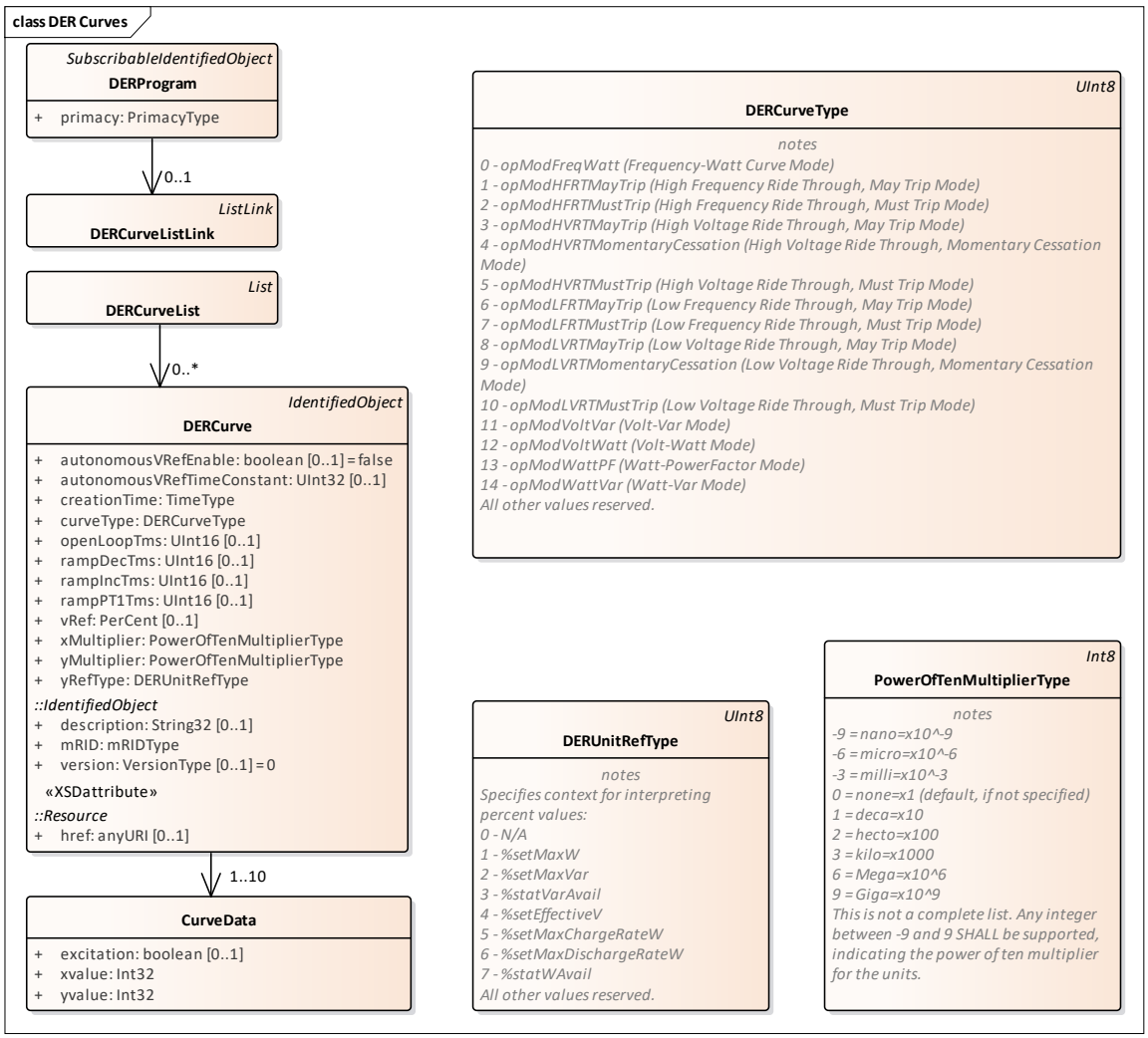


Figure B.36—DER curves

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

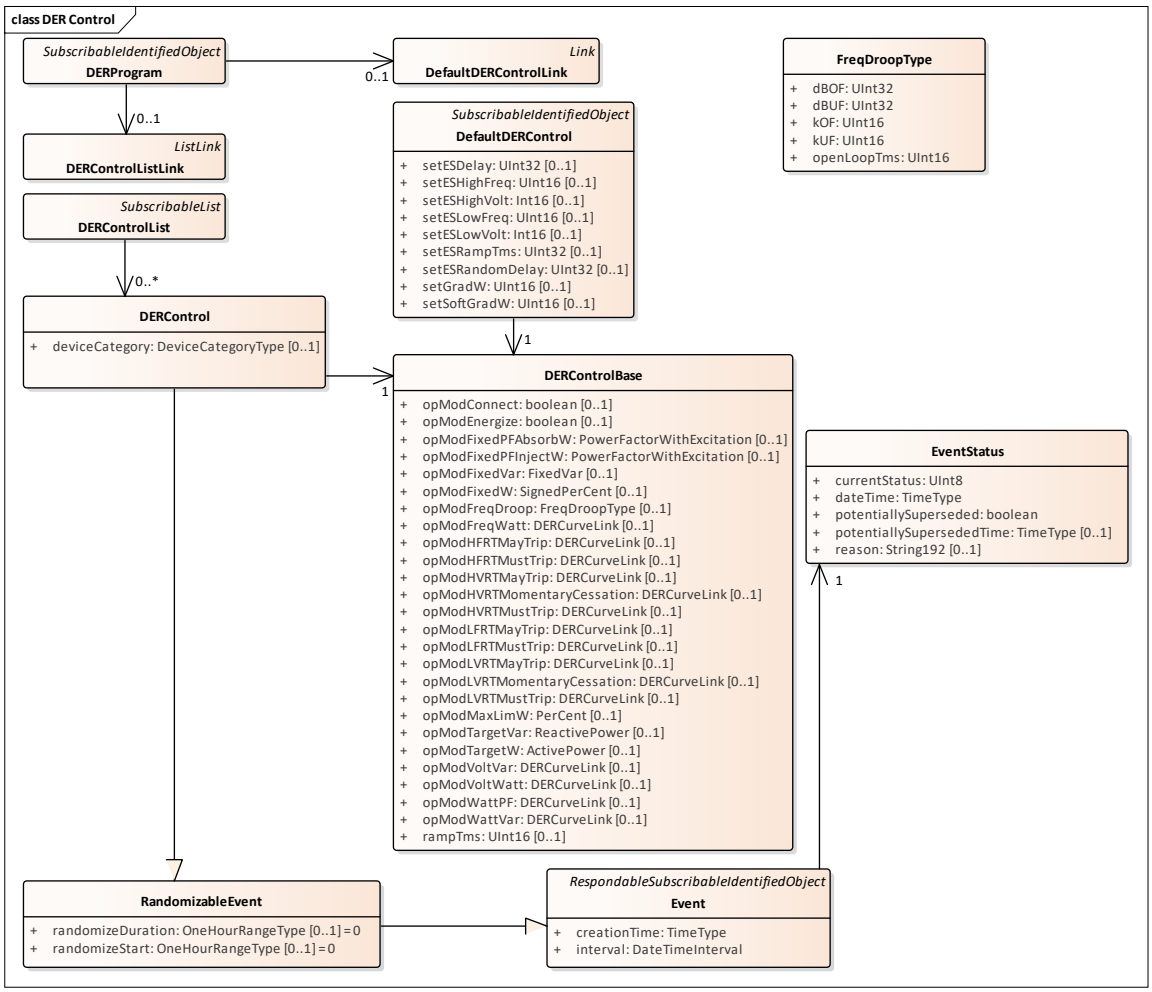


Figure B.37—DER control

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

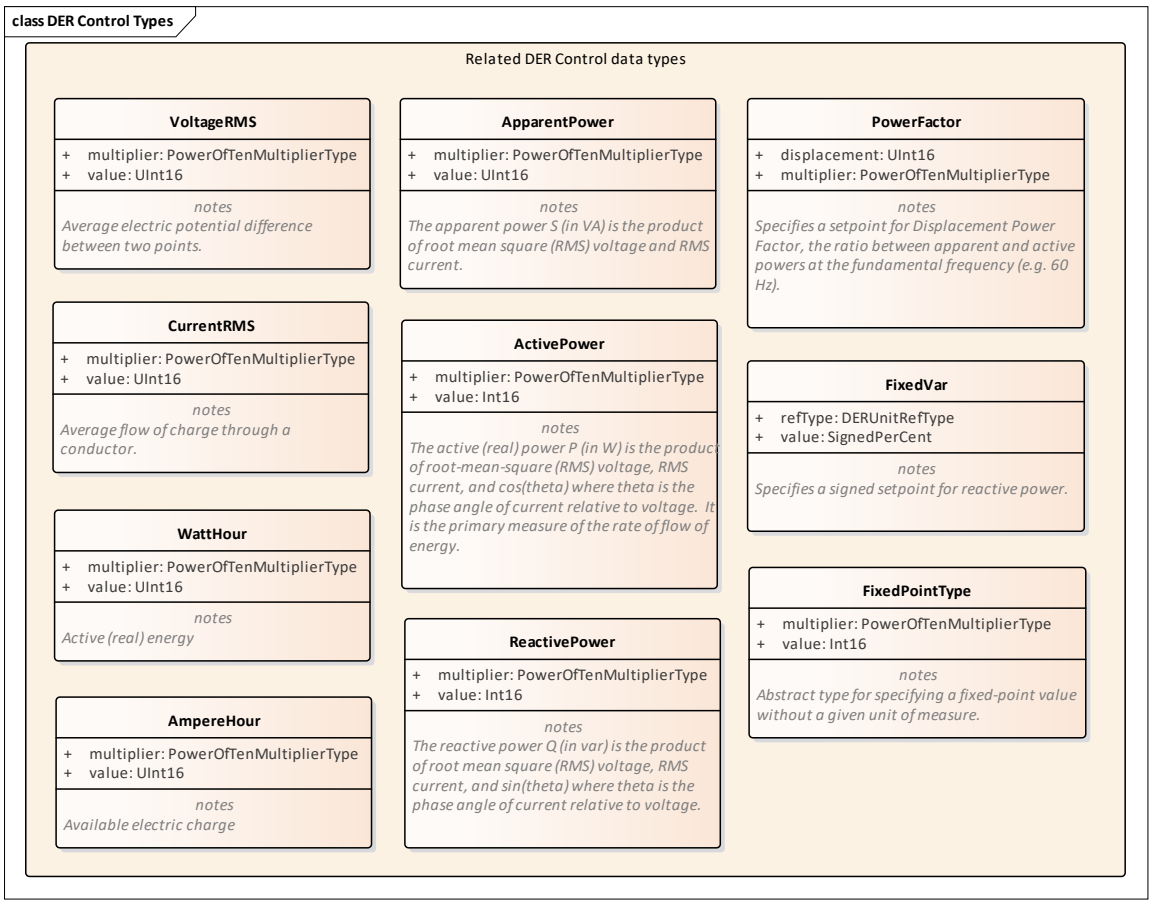


Figure B.38—DER control types

DefaultDERControl object (SubscribableIdentifiedObject)

Contains control mode information to be used if no active DERControl is found.

setESDelay attribute (UInt32) [0..1]

Enter service delay, in hundredths of a second. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESDelay).

setESHighFreq attribute (UInt16) [0..1]

Enter service frequency high. Specified in hundredths of hertz. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESHighFreq).

setESHighVolt attribute (Int16) [0..1]

Enter service voltage high. Specified as an effective percent voltage, defined as $100\% \times \frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$, in hundredths of a percent. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESHighVolt).

setESLowFreq attribute (UInt16) [0..1]

Enter service frequency low. Specified in hundredths of hertz. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESLowFreq).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

setESLowVolt attribute (Int16) [0..1]
Enter service voltage low. Specified as an effective percent voltage, defined as $100\% \times \frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$, in hundredths of a percent. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESLowVolt).

setESRampTms attribute (UInt32) [0..1]
Enter service ramp time, in hundredths of a second. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESRampTms).

setESRandomDelay attribute (UInt32) [0..1]
Enter service randomized delay, in hundredths of a second. When present, this value SHALL update the value of the corresponding setting (DERSettings::setESRandomDelay).

setGradW attribute (UInt16) [0..1]
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. A value of 0 means there is no limit. Interpreted as a percentage change in output capability limit per second when used as a default ramp rate. When present, this value SHALL update the value of the corresponding setting (DERSettings::setGradW).

setSoftGradW attribute (UInt16) [0..1]
Set soft-start rate of change (soft-start ramp rate) of active power output due to command or internal action, defined in %setWMax/second. Resolution is in hundredths of a percent/second. A value of 0 means there is no limit. Interpreted as a percentage change in output capability limit per second when used as a ramp rate. When present, this value SHALL update the value of the corresponding setting (DERSettings::setSoftGradW).

FreqDroopType object ()
Type for Frequency-Droop (Frequency-Watt) operation.

dBOF attribute (UInt32)
Frequency droop dead band for over-frequency conditions. In thousandths of hertz.

dBUF attribute (UInt32)
Frequency droop dead band for under-frequency conditions. In thousandths of hertz.

kOF attribute (UInt16)
Frequency droop per-unit frequency change for over-frequency conditions corresponding to one per-unit power output change. In thousandths, unitless.

kUF attribute (UInt16)
Frequency droop per-unit frequency change for under-frequency conditions corresponding to one per-unit power output change. In thousandths, unitless.

openLoopTms attribute (UInt16)
Open loop response time, the duration from a step change in control signal input until the output changes by 90% of its final change before any overshoot, in hundredths of a second. Resolution is 1/100 second. A value of 0 is used to mean no limit.

DER object (SubscribableResource)
Contains links to DER resources.

DERList object (List)
A List element to hold DER objects.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

pollRate attribute (UInt32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

DERSettings object (SubscribableResource)
Distributed energy resource settings.

modesEnabled attribute (DERControlType) [0..1]
Bitmap indicating the DER Controls enabled on the device. See DERControlType for values. If a control is supported (see DERCapability::modesSupported), but not enabled, the control will not be executed if encountered.

setESDelay attribute (UInt32) [0..1]
Enter service delay, in hundredths of a second.

setESHighFreq attribute (UInt16) [0..1]
Enter service frequency high. Specified in hundredths of hertz.

setESHighVolt attribute (Int16) [0..1]
Enter service voltage high. Specified as an effective percent voltage, defined as $100\% \times \frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$, in hundredths of a percent.

setESLowFreq attribute (UInt16) [0..1]
Enter service frequency low. Specified in hundredths of hertz.

setESLowVolt attribute (Int16) [0..1]
Enter service voltage low. Specified as an effective percent voltage, defined as $100\% \times \frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$, in hundredths of a percent.

setESRampTms attribute (UInt32) [0..1]
Enter service ramp time, in hundredths of a second.

setESRandomDelay attribute (UInt32) [0..1]
Enter service randomized delay, in hundredths of a second.

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. A value of 0 means there is no limit. Interpreted as a percentage change in output capability limit per second when used as a default ramp rate.

setMaxA attribute (CurrentRMS) [0..1]
AC current maximum. Maximum ac current in rms amperes.

setMaxAh attribute (AmpereHour) [0..1]
Maximum usable energy storage capacity of the DER, in ampere-hours. Note: this may be different from physical capability.

setMaxChargeRateVA attribute (ApparentPower) [0..1]
Apparent power charge maximum. Maximum apparent power the DER can absorb from the grid in voltamperes. May differ from the apparent power maximum (setMaxVA).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

setMaxChargeRateW attribute (ActivePower) [0..1]
Maximum rate of energy transfer received by the storage device, in watts. Defaults to rtgMaxChargeRateW.

setMaxDischargeRateVA attribute (ApparentPower) [0..1]
Apparent power discharge maximum. Maximum apparent power the DER can deliver to the grid in voltamperes. May differ from the apparent power maximum (setMaxVA).

setMaxDischargeRateW attribute (ActivePower) [0..1]
Maximum rate of energy transfer delivered by the storage device, in watts. Defaults to rtgMaxDischargeRateW.

setMaxV attribute (VoltageRMS) [0..1]
AC voltage maximum setting.

setMaxVA attribute (ApparentPower) [0..1]
Set limit for maximum apparent power capability of the DER (in VA). Defaults to rtgMaxVA.

setMaxVar attribute (ReactivePower) [0..1]
Set limit for maximum reactive power delivered by the DER (in var). SHALL be a positive value ≤ rtgMaxVar (default).

setMaxVarNeg attribute (ReactivePower) [0..1]
Set limit for maximum reactive power received by the DER (in var). If present, SHALL be a negative value ≥ rtgMaxVarNeg (default). If absent, defaults to negative setMaxVar.

setMaxW attribute (ActivePower)
Set limit for maximum active power capability of the DER (in watts). Defaults to rtgMaxW.

setMaxWh attribute (WattHour) [0..1]
Maximum energy storage capacity of the DER, in watthours. Note: this may be different from physical capability.

setMinPFOverExcited attribute (PowerFactor) [0..1]
Set minimum power factor displacement limit of the DER when injecting reactive power (over-excited); SHALL be a positive value between 0.0 (typically > 0.7) and 1.0. SHALL be ≥ rtgMinPFOverExcited (default).

setMinPFUnderExcited attribute (PowerFactor) [0..1]
Set minimum power factor displacement limit of the DER when absorbing reactive power (under-excited); SHALL be a positive value between 0.0 (typically > 0.7) and 0.9999. If present, SHALL be ≥ rtgMinPFUnderExcited (default). If absent, defaults to setMinPFOverExcited.

setMinV attribute (VoltageRMS) [0..1]
AC voltage minimum setting.

setSoftGradW attribute (UInt16) [0..1]
Set soft-start rate of change (soft-start ramp rate) of active power output due to command or internal action, defined in %setWMax/second. Resolution is in hundredths of a percent/second. A value of 0 means there is no limit. Interpreted as a percentage change in output capability limit per second when used as a ramp rate.

setVNom attribute (VoltageRMS) [0..1]
AC voltage nominal setting.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

setVRef attribute (VoltageRMS) [0..1]
The nominal ac voltage (rms) at the utility's point of common coupling.

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.

updatedTime attribute (TimeType)
Specifies the time at which the DER information was last updated.

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

6 = Other generation system

80 = Other storage system

81 = Electric vehicle

82 = EVSE

83 = Combined PV and storage

All other values reserved.

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 (%setMaxChargeRateW) 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

rtgAbnormalCategory attribute (UInt8) [0..1]
Abnormal operating performance category as defined by IEEE Std 1547-2018. One of:

0 = Not specified

1 = Category I

2 = Category II

3 = Category III

All other values reserved.

rtgMaxA attribute (CurrentRMS) [0..1]
Maximum continuous ac current capability of the DER, in amperes (rms).

rtgMaxAh attribute (AmpereHour) [0..1]
Usable energy storage capacity of the DER, in amp hours.

rtgMaxChargeRateVA attribute (ApparentPower) [0..1]
Maximum apparent power charge rating in voltamperes. May differ from the maximum apparent power rating.

rtgMaxChargeRateW attribute (ActivePower) [0..1]
Maximum rate of energy transfer received by the storage DER, in watts.

rtgMaxDischargeRateVA attribute (ApparentPower) [0..1]
Maximum apparent power discharge rating in voltamperes. May differ from the maximum apparent power rating.

rtgMaxDischargeRateW 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 == 83).

rtgMaxV attribute (VoltageRMS) [0..1]
AC voltage maximum rating.

rtgMaxVA attribute (ApparentPower) [0..1]
Maximum continuous apparent power output capability of the DER, in voltamperes.

rtgMaxVar attribute (ReactivePower) [0..1]
Maximum continuous reactive power delivered by the DER, in var.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

rtgMaxVarNeg attribute (ReactivePower) [0..1]
Maximum continuous reactive power received by the DER, in var. If absent, defaults to negative rtgMaxVar.

rtgMaxW attribute (ActivePower)
Maximum continuous active power output capability of the DER, in watts. Represents combined generation plus storage output if DERType == 83.

rtgMaxWh attribute (WattHour) [0..1]
Maximum energy storage capacity of the DER, in watthours.

rtgMinPFOverExcited attribute (PowerFactor) [0..1]
Minimum power factor displacement capability of the DER when injecting reactive power (over-excited); SHALL be a positive value between 0.0 (typically > 0.7) and 1.0. If absent, defaults to unity.

rtgMinPFUnderExcited attribute (PowerFactor) [0..1]
Minimum power factor displacement capability of the DER when absorbing reactive power (under-excited); SHALL be a positive value between 0.0 (typically > 0.7) and 0.9999. If absent, defaults to rtgMinPFOverExcited.

rtgMinV attribute (VoltageRMS) [0..1]
AC voltage minimum rating.

rtgNormalCategory attribute (UInt8) [0..1]
Normal operating performance category as defined by IEEE Std 1547-2018. One of:

0 = Not specified

1 = Category A

2 = Category B

All other values reserved.

rtgOverExcitedPF attribute (PowerFactor) [0..1]
Specified over-excited power factor.

rtgOverExcitedW attribute (ActivePower) [0..1]
Active power rating in watts at specified over-excited power factor (rtgOverExcitedPF). If present, rtgOverExcitedPF SHALL be present.

rtgReactiveSusceptance attribute (ReactiveSusceptance) [0..1]
Reactive susceptance that remains connected to the area electric power system (EPS) in the cease to energize and trip state.

rtgUnderExcitedPF attribute (PowerFactor) [0..1]
Specified under-excited power factor.

rtgUnderExcitedW attribute (ActivePower) [0..1]
Active power rating in watts at specified under-excited power factor (rtgUnderExcitedPF). If present, rtgUnderExcitedPF SHALL be present.

rtgVNom attribute (VoltageRMS) [0..1]
AC voltage nominal rating.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

type attribute (DERType)
Type of DER; see DERType object.

DERControlBase object ()
Distributed energy resource (DER) control values.

opModConnect attribute (boolean) [0..1]
Set DER as connected (true) or disconnected (false). Used in conjunction with ramp rate when re-connecting. Implies galvanic isolation.

opModEnergize attribute (boolean) [0..1]
Set DER as energized (true) or de-energized (false). Used in conjunction with ramp rate when re-energizing.

opModFixedPFAbsorbW attribute (PowerFactorWithExcitation) [0..1]
The opModFixedPFAbsorbW function specifies a requested fixed power factor (PF) setting for when active power is being absorbed. The actual displacement SHALL be within the limits established by setMinPFOverExcited and setMinPFUnderExcited. If issued simultaneously with other reactive power controls (e.g., opModFixedVar) the control resulting in least var magnitude SHOULD take precedence.

opModFixedPFInjectW attribute (PowerFactorWithExcitation) [0..1]
The opModFixedPFInjectW function specifies a requested fixed power factor (PF) setting for when active power is being injected. The actual displacement SHALL be within the limits established by setMinPFOverExcited and setMinPFUnderExcited. If issued simultaneously with other reactive power controls (e.g., opModFixedVar) the control resulting in least var magnitude SHOULD take precedence.

opModFixedVar attribute (FixedVar) [0..1]
The opModFixedVar function specifies the delivered or received reactive power setpoint. The context for the setpoint value is determined by refType and SHALL be one of %setMaxW, %setMaxVar, or %statVarAvail. If issued simultaneously with other reactive power controls (e.g., opModFixedPFInjectW) the control resulting in least var magnitude SHOULD take precedence.

opModFixedW attribute (SignedPerCent) [0..1]
The opModFixedW function specifies a requested charge or discharge mode setpoint, in %setMaxChargeRateW if negative value, or %setMaxW or %setMaxDischargeRateW if positive value (in hundredths).

opModFreqDroop attribute (FreqDroopType) [0..1]
Specifies a frequency-watt operation. This operation limits active power generation or consumption when the line frequency deviates from nominal by a specified amount.

opModFreqWatt attribute (DERCurveLink) [0..1]
Specify DERCurveLink for curveType == 0. 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 hertz. The y value specifies a corresponding active power output in %setMaxW.

opModHFRTMayTrip attribute (DERCurveLink) [0..1]
Specify DERCurveLink for curveType == 1. The High Frequency Ride-Through (HFRT) function is specified by one or two duration-frequency curves that define the operating region under high-frequency conditions. Each HFRT curve is specified by an array of duration-frequency 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 frequency in seconds). The y value of each pair specifies a frequency, in hertz. This control specifies the “may trip” region.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

opModHFRTMustTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 2. The High Frequency Ride-Through (HFRT) function is specified by a duration-frequency curve that defines the operating region under high-frequency conditions. Each HFRT curve is specified by an array of duration-frequency 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 frequency in seconds). The y value of each pair specifies a frequency, in hertz. This control specifies the "must trip" region.

opModHVRTMayTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 3. The High Voltage Ride-Through (HVRT) function is specified by one, two, or three 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 percentage voltage, defined as $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$. This control specifies the "may trip" region.

opModHVRTMomentaryCessation attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 4. The High Voltage Ride-Through (HVRT) function is specified by 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 $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$. This control specifies the "momentary cessation" region.

opModHVRTMustTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 5. The High Voltage Ride-Through (HVRT) function is specified by 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 $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$. This control specifies the "must trip" region.

opModLFRTMayTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 6. The Low Frequency Ride-Through (LFRT) function is specified by one or two duration-frequency curves that define the operating region under low-frequency conditions. Each LFRT curve is specified by an array of duration-frequency 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 frequency in seconds). The y value of each pair specifies a frequency, in hertz. This control specifies the "may trip" region.

opModLFRTMustTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 7. The Low Frequency Ride-Through (LFRT) function is specified by a duration-frequency curve that defines the operating region under low-frequency conditions. Each LFRT curve is specified by an array of duration-frequency 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 frequency in seconds). The y value of each pair specifies a frequency, in hertz. This control specifies the "must trip" region.

opModLVRTMayTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 8. The Low Voltage Ride-Through (LVRT) function is specified by one, two, or three duration-volt curves that define the operating region under low-voltage conditions.

Copyright © 2018 IEEE. All rights reserved.

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 $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$. This control specifies the “may trip” region.

opModLVRTMomentaryCessation attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 9. The Low Voltage Ride-Through (LVRT) function is specified by 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 $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$. This control specifies the “momentary cessation” region.

opModLVRTMustTrip attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 10. The Low Voltage Ride-Through (LVRT) function is specified by 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 $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$. This control specifies the “must trip” region.

opModMaxLimW attribute (PerCent) [0..1]

The opModMaxLimW 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, for example, by reducing PV output or by using excess PV output to charge associated storage.

opModTargetVar attribute (ReactivePower) [0..1]

Target reactive power, in var. This control is likely to be more useful for aggregators, as individual DERs may not be able to maintain a target setting.

opModTargetW attribute (ActivePower) [0..1]

Target output power, in watts. This control is likely to be more useful for aggregators, as individual DERs may not be able to maintain a target setting.

opModVoltVar attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 11. 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 $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$ and SHOULD support a domain of at least 0 to 135. If VRef is present in DERCurve, then the x value of each pair is additionally multiplied by $\frac{\text{VRef}}{10\,000}$. The y value specifies a target var output interpreted as a signed percentage (−100 to 100). The meaning of the y value is determined by yRefType and must be one of %setMaxW, %setMaxVar, or %statVarAvail.

opModVoltWatt attribute (DERCurveLink) [0..1]

Specify DERCurveLink for curveType == 12. 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,

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

defined as $\frac{(\text{locally measured voltage} - \text{setVRefOfs})}{\text{setVRef}}$ and SHOULD support a domain of at least 0 to 135. The y value specifies an active power output interpreted as a percentage (0 to 100). The meaning of the y value is determined by yRefType and must be one of %setMaxW or %statWAvail.

opModWattPF attribute (DERCurveLink) [0..1]
Specify DERCurveLink for curveType == 13. 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 to 100). 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., opModFixedPFInjectW) the control resulting in least var magnitude SHOULD take precedence.

opModWattVar attribute (DERCurveLink) [0..1]
Specify DERCurveLink for curveType == 14. The Watt-Var function varies vars as a function of delivered active power. The Watt-Var curve is specified as an array of Watt-Var pairs that are interpolated into a piecewise linear function with hysteresis. The x value of each pair specifies a watt setting in %setMaxW, (0 to 100). The y value specifies a target var output interpreted as a signed percentage (−100 to 100). The meaning of the y value is determined by yRefType and must be one of %setMaxW, %setMaxVar, or %statVarAvail.

rampTms attribute (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 (setGradW). Resolution is 1/100 sec.

DERControl object (RandomizableEvent)
Distributed energy resource (DER) time/event-based control.

deviceCategory attribute (DeviceCategoryType) [0..1]
Specifies the bitmap indicating the categories of devices that SHOULD respond. Devices SHOULD ignore events that do not indicate their device category. If not present, all devices SHOULD respond.

DERControlList object (SubscribableList)
A List element to hold DERControl objects.

DERControlType object (HexBinary32)
Control modes supported by the DER. Bit positions SHALL be defined as follows:

0 = Charge mode

1 = Discharge mode

2 = opModConnect (connect/disconnect—implies galvanic isolation)

3 = opModEnergize (energize/de-energize)

4 = opModFixedPFAbsorbW (fixed power factor setpoint when absorbing active power)

5 = opModFixedPFInjectW (fixed power factor setpoint when injecting active power)

6 = opModFixedVar (reactive power setpoint)

Copyright © 2018 IEEE. All rights reserved.

7 = opModFixedW (charge/discharge setpoint)

8 = opModFreqDroop (Frequency-Watt Parameterized mode)

9 = opModFreqWatt (Frequency-Watt Curve mode)

10 = opModHFRTMayTrip (High Frequency Ride-Through, May Trip mode)

11 = opModHFRTMustTrip (High Frequency Ride-Through, Must Trip mode)

12 = opModHVRTMayTrip (High Voltage Ride-Through, May Trip mode)

13 = opModHVRTMomentaryCessation (High Voltage Ride-Through, Momentary Cessation mode)

14 = opModHVRTMustTrip (High Voltage Ride-Through, Must Trip mode)

15 = opModLFRTMayTrip (Low Frequency Ride-Through, May Trip mode)

16 = opModLFRTMustTrip (Low Frequency Ride-Through, Must Trip mode)

17 = opModLVRTMayTrip (Low Voltage Ride-Through, May Trip mode)

18 = opModLVRTMomentaryCessation (Low Voltage Ride-Through, Momentary Cessation mode)

19 = opModLVRTMustTrip (Low Voltage Ride-Through, Must Trip mode)

20 = opModMaxLimW (maximum active power)

21 = opModTargetVar (target reactive power)

22 = opModTargetW (target active power)

23 = opModVoltVar (Volt-Var mode)

24 = opModVoltWatt (Volt-Watt mode)

25 = opModWattPF (Watt-Powerfactor mode)

26 = opModWattVar (Watt-Var mode)

All other values reserved.

DERCurve object (IdentifiedObject)

DER related curves such as Volt-Var mode curves. Relationship between an independent variable (X-axis) and a dependent variable (Y-axis).

autonomousVRefEnable attribute (boolean) [0..1]

If the curveType is opModVoltVar, then this field MAY be present. If the curveType is not opModVoltVar, then this field SHALL NOT be present. Enable/disable autonomous vRef adjustment. When enabled, the Volt-Var curve characteristic SHALL be adjusted autonomously as vRef changes and autonomousVRefTimeConstant SHALL be present. If a DER is able to support Volt-Var mode but is unable to support autonomous vRef adjustment, then the DER SHALL execute the curve without autonomous vRef adjustment. If not specified, then the value is false.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

autonomousVRefTimeConstant attribute (UInt32) [0..1]
If the curveType is opModVoltVar, then this field MAY be present. If the curveType is not opModVoltVar, then this field SHALL NOT be present. Adjustment range for vRef time constant, in hundredths of a second.

creationTime attribute (TimeType)
The time at which the object was created.

curveType attribute (DERCurveType)
Specifies the associated curve-based control mode.

openLoopTms attribute (UInt16) [0..1]
Open loop response time, the time to ramp up to 90% of the new target in response to the change in voltage, in hundredths of a second. Resolution is 1/100 second. A value of 0 is used to mean no limit. When not present, the device SHOULD follow its default behavior.

rampDecTms attribute (UInt16) [0..1]
Decreasing 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. A value of 0 means there is no limit. If absent, ramp rate defaults to setGradW.

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. A value of 0 means there is no limit. If absent, ramp rate defaults to rampDecTms.

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 second.

vRef attribute (PerCent) [0..1]
If the curveType is opModVoltVar, then this field MAY be present. If the curveType is not opModVoltVar, then this field SHALL NOT be present. The nominal ac voltage (rms) adjustment to the voltage curve points for Volt-Var curves.

xMultiplier attribute (PowerOfTenMultiplierType)
Exponent for X-axis value.

yMultiplier attribute (PowerOfTenMultiplierType)
Exponent for Y-axis value.

yRefType attribute (DERUnitRefType)
The Y-axis units context.

CurrentDERProgramLink object (Link)
SHALL contain a Link to an instance of DERProgram. If present, this is the DERProgram containing the currently active DERControl.

DERCurveList object (List)
A List element to hold DERCurve objects.

CurveData object ()
Data point values for defining a curve or schedule.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

excitation attribute (boolean) [0..1]
If yvalue is power factor, then this field SHALL be present. If yvalue is not power factor, then this field SHALL NOT be present.

True when DER is absorbing reactive power (under-excited), false when DER is injecting reactive power (over-excited).

xvalue attribute (Int32)
The data value of the X-axis (independent) variable, depending on the curve type. See definitions in DERControlBase for further information.

yvalue attribute (Int32)
The data value of the Y-axis (dependent) variable, depending on the curve type. See definitions in DERControlBase for further information. If yvalue is power factor, the excitation field SHALL be present and yvalue SHALL be a positive value. If yvalue is not power factor, the excitation field SHALL NOT be present.

DERCurveType object (UInt8)
0 = opModFreqWatt (Frequency-Watt Curve mode)

1 = opModHFRTMayTrip (High Frequency Ride-Through, May Trip mode)

2 = opModHFRTMustTrip (High Frequency Ride-Through, Must Trip mode)

3 = opModHVRTMayTrip (High Voltage Ride-Through, May Trip mode)

4 = opModHVRTMomentaryCessation (High Voltage Ride-Through, Momentary Cessation mode)

5 = opModHVRTMustTrip (High Voltage Ride-Through, Must Trip mode)

6 = opModLFRTMayTrip (Low Frequency Ride-Through, May Trip mode)

7 = opModLFRTMustTrip (Low Frequency Ride-Through, Must Trip mode)

8 = opModLVRTMayTrip (Low Voltage Ride-Through, May Trip mode)

9 = opModLVRTMomentaryCessation (Low Voltage Ride-Through, Momentary Cessation mode)

10 = opModLVRTMustTrip (Low Voltage Ride-Through, Must Trip mode)

11 = opModVoltVar (Volt-Var mode)

12 = opModVoltWatt (Volt-Watt mode)

13 = opModWattPF (Watt-PowerFactor mode)

14 = opModWattVar (Watt-Var mode)

All other values reserved.

DERProgram object (SubscribableIdentifiedObject)
Distributed energy resource program.

primacy attribute (PrimacyType)
Indicates the relative primacy of the provider of this program.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

DERProgramList object (SubscribableList)
A List element to hold DERProgram objects.

pollRate attribute (Uint32) [0..1] «XSDattribute»
The default polling rate for this function set (this resource and all resources below), in seconds. If not specified, a default of 900 seconds (15 minutes) is used. It is RECOMMENDED that a client poll the resources of this function set every pollRate seconds.

DERStatus object (SubscribableResource)
DER status information.

alarmStatus attribute (HexBinary32) [0..1]
Bitmap indicating the status of DER alarms (see DER LogEvents for more details).

0 = DER_FAULT_OVER_CURRENT

1 = DER_FAULT_OVER_VOLTAGE

2 = DER_FAULT_UNDER_VOLTAGE

3 = DER_FAULT_OVER_FREQUENCY

4 = DER_FAULT_UNDER_FREQUENCY

5 = DER_FAULT_VOLTAGE_IMBALANCE

6 = DER_FAULT_CURRENT_IMBALANCE

7 = DER_FAULT_EMERGENCY_LOCAL

8 = DER_FAULT_EMERGENCY_REMOTE

9 = DER_FAULT_LOW_POWER_INPUT

10 = DER_FAULT_PHASE_ROTATION

11 to 31 = Reserved

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 = %setMaxChargeRateW

6 = %setMaxDischargeRateW

7 = %statWAvail

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.

Copyright © 2018 IEEE. All rights reserved.

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 watts) 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 voltamperes) is the product of root mean square (rms) voltage and rms current.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of uom.

value attribute (UInt16)
Value in voltamperes (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 voltamperes reactive (var) (uom 63).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

ReactiveSusceptance object ()
Reactive susceptance.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of uom.

value attribute (UInt16)
Value in siemens (uom 53).

PowerFactor 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 (UInt16)
Significand of an unsigned value of cos(theta) between 0 and 1.0. For example, a value of 0.95 may be specified as a displacement of 950 and a multiplier of −3.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of "displacement."

PowerFactorWithExcitation object ()
Specifies a setpoint for displacement power factor, the ratio between apparent and active powers at the fundamental frequency (e.g., 60 Hz) and includes an excitation flag.

displacement attribute (UInt16)
Significand of an unsigned value of cos(theta) between 0 and 1.0. For example, a value of 0.95 may be specified as a displacement of 950 and a multiplier of −3.

excitation attribute (boolean)
True when DER is absorbing reactive power (under-excited), false when DER is injecting reactive power (over-excited).

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.

Copyright © 2018 IEEE. All rights reserved.

multiplier attribute (PowerOfTenMultiplierType)
Specifies exponent of uom.

value attribute (UInt16)
Value in volts rms (uom 29).

ConnectStatusType object ()
DER ConnectStatus value (bitmap):

0 = Connected

1 = Available

2 = Operating

3 = Test

4 = Fault/Error

All other values reserved.

dateTime attribute (TimeType)
The date and time at which the state applied.

value attribute (HexBinary8)
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 maximum power point 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

B.2.23 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.

ActiveSupplyInterruptionOverrideListLink object (ListLink)
SHALL contain a Link to a List of active SupplyInterruptionOverride 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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 sub-meter 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Copyright © 2018 IEEE. All rights reserved.

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.

LoadShedAvailabilityListLink object (ListLink)
SHALL contain a Link to a List of LoadShedAvailability instances.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Annex C

(informative)

Examples and guideline

C.1 Introduction

The contents of this annex are not considered to be normative and are provided for reference. Text contained in { } in the following examples is considered to be placeholders for the actual text or value.

C.2 Registration: Remote

The flow diagram in Figure C.1 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.

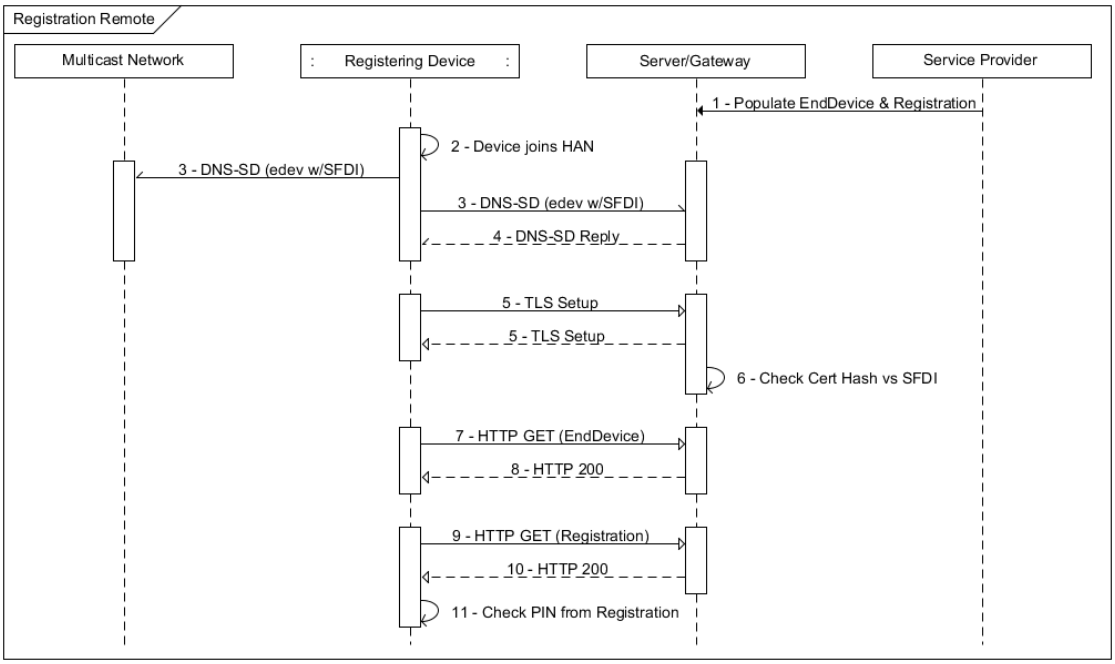


Figure C.1—Remote registration

NOTE—An XML example in POX encoding is provided along with the EXI equivalent.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.1—POX example: Registration remote

Step	Description
1	(Out of band) Service provider populates client device's EndDevice (containing short form device identifier [SFDI]) and Registration (containing personal identification number [PIN]) resources to appropriate HAN registration server (typically the ESI).
2	Client device joins the HAN (layer 2).
3	Client issues DNS-SD request to locate its EndDevice (keyed by its SFDI).
4	A server or multiple servers provides DNS-SD responses with URL to client's EndDevice. If no reply is found, then there are no specific registrations for this device on this network.
5	For each reply received the client performs TLS and client authentication and executes the following steps. Note that client certificate is sent in the clear and thus SFDI can be determined.
6	Client and server now have an encrypted connection. Each has determined it is talking to an authenticated IEEE 2030.5 device, but have not confirmed they are talking to the correct specific IEEE 2030.5 device.
7	Server verifies client identity because client certificate hashes to the client SFDI.
8	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+xml`
9	Server responds with the EndDevice resource. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<EndDevice href="/edev/3" subscribable="0" xmlns="urn:ieee:std:2030.5:ns">` `  <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>` `  <changedTime>1379905200</changedTime>` `  <FunctionSetAssignmentsListLink all="3" href="/edev/3/fsal"/>` `  <RegistrationLink href="/edev/3/reg"/>` `  <SubscriptionListLink all="0" href="/edev/3/subl"/>` `</EndDevice>`
10	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+xml`
11	Server responds with the Registration resource. Note: the PIN is thus transmitted over a secure channel. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<Registration href="/edev/3/reg" xmlns="urn:ieee:std:2030.5:ns">` `  <dateTimeRegistered>1364774400</dateTimeRegistered>` `  <pIN>123455</pIN>` `</Registration>`
12	Client verifies its PIN versus that provided by the server. If the PIN is found to be invalid, then the client knows it is not registered with this server.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.2—EXI example: Registration remote

Step	Description
1	(Same as POX example)
2	(Same as POX example)
3	(Same as POX example)
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=+S1` Note: level=+S1 indicates the server can send/receive EXI with the IEEE 2030.5-2018 schema with arbitrary extensions, so subsequent HTTP content on the server may contain extended parts (not used in IEEE Std 2030.5-2018 but may be defined in future versions of IEEE Std 2030.5).
5	(Same as POX example)
6	(Same as POX example)
7	Client GETs its EndDevice from the 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=-S1` Note that `level=-S1` indicates the client can receive EXI with the IEEE 2030.5-2018 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 c66e 0092 f656 4657 62f3 3000` `0069 7b2b 232b b179 997b 1b33 3881 85ec` `ac8c aec5 e665 ec8d 2006 17b2 b232 bb17` `9997 b239 8030 bd95 9195 d8bc ccbd 99cd` `0185 ecac 8cae c5e6 65ee 0e63 b79a 8e9b` `7c70 2c37 7fa4 4146 0183 8bd9 5919 5d8b` `cccb d99c d85b 041a 5eca c8ca ec5e 665e` `e4ca ce00 0071 7b2b 232b b179 997b 9bab` `1360` (130 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=-S1`
10	Server responds with the Registration resource. Note that the PIN is thus transmitted over a secure channel. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep-exi` `Content-Length: {contentLength}` `a030 114c c6f7 034b d959 195d 8bcc cbdc` `9959 d202 738e 2814 5fe2 0380` (28 bytes in binary, shown in hexadecimal)
11	(Same as POX example)

Copyright © 2018 IEEE. All rights reserved.

C.3 Registration: Local

The flow diagram shown in Figure C.2 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.

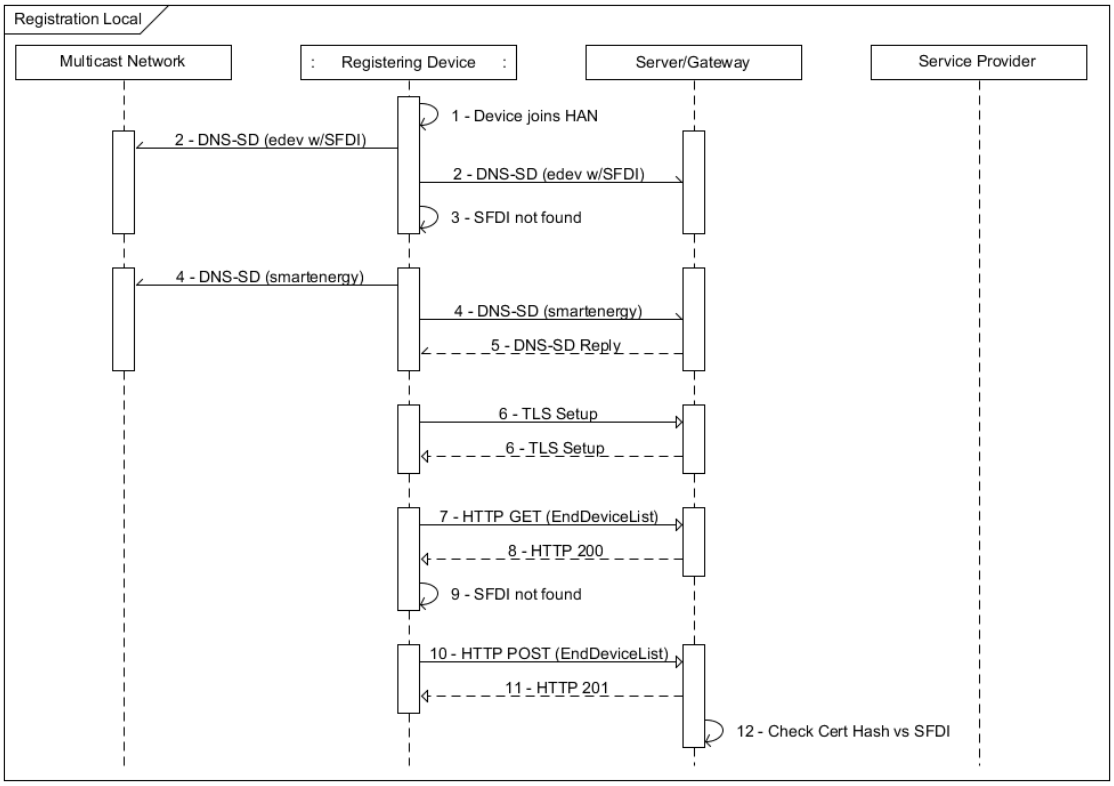


Figure C.2—Registration local

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.3—POX example: Registration local

Step	Description
1	Client device joins the HAN (layer 2).
2	Client issues DNS-SD request to locate its EndDevice (keyed by its SFDI).
3	Client does not find a desired SFDI response.
4	Client issues DNS-SD request to locate any “smartenergy” server.
5	A Server or multiple servers provides DNS-SD responses. If no reply is found, then no “smartenergy” servers are present on the network.
6	For each reply received the client performs TLS and client authentication and executes the following steps. Note that client certificate is sent in the clear and thus SFDI can be determined. Client and server now have an encrypted connection. Each has determined it is talking to an authenticated IEEE 2030.5 device, but have not confirmed they are talking to the correct specific IEEE 2030.5 device.
7	Client GETs the EndDeviceList from Server as returned in DNS-SD Discovery. Client sends the following request: `GET /edev HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`
8	Server responds with the EndDeviceList resource. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<EndDeviceList all="1" href="/edev" results="1" subscribable="0"` `xmlns="urn:ieee:std:2030.5:ns">` `  <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>` `  <changedTime>1379905200</changedTime>` `  <FunctionSetAssignmentsListLink all="3" href="/edev/3/fsal"/>` `  <RegistrationLink href="/edev/3/reg"/>` `  <SubscriptionListLink all="0" href="/edev/3/subl"/>` `  </EndDevice>` `</EndDeviceList>`
9	Client does not find its SFDI in any of the EndDevices in the EndDeviceList.
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 xmlns="urn:ieee:std:2030.5:ns">` `  <sFDI>789654321005</sFDI>` `  <changedTime>1379905200</changedTime>` `</EndDevice>`
11	Server indicates the EndDevice record was created with the following: If a 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 the client’s identity because the client certificate hashes to the client SFDI.

Copyright © 2018 IEEE. All rights reserved.

C.4 Discovery: Function Set Assignment

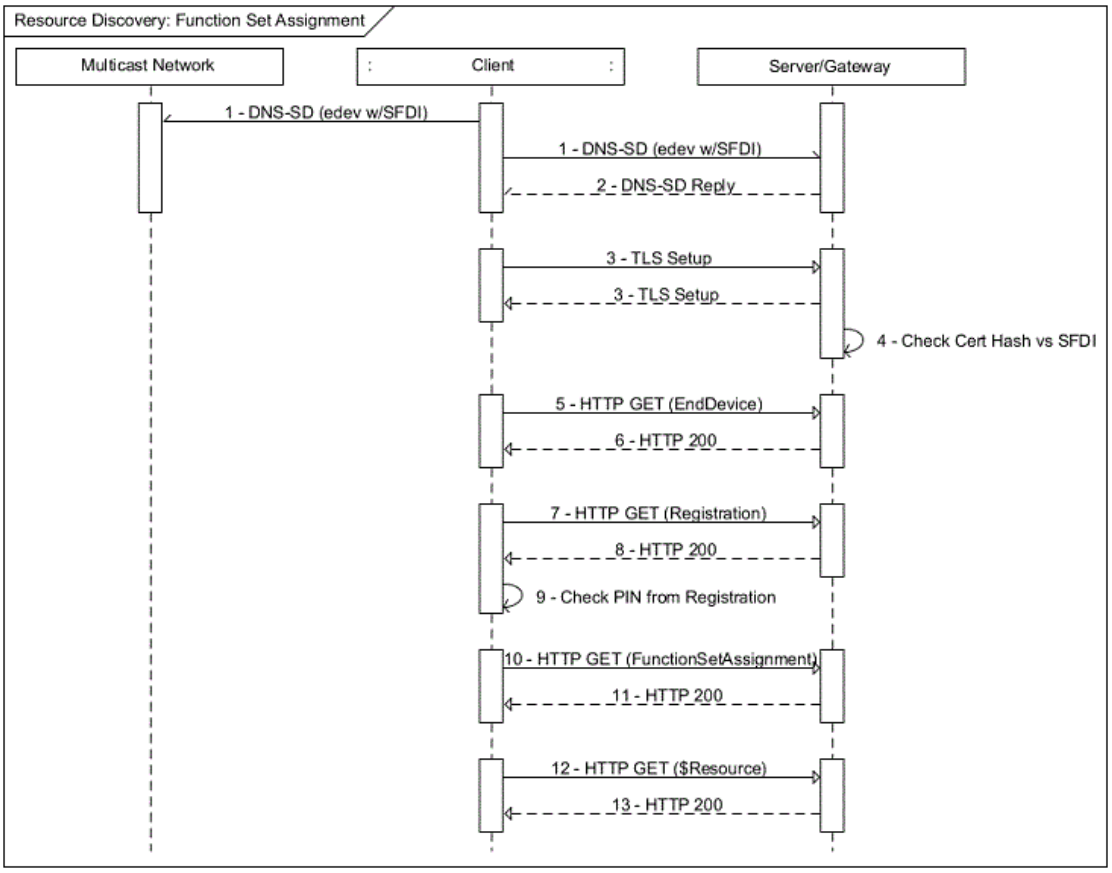


Figure C.3—Discovery: Function Set Assignment

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.4—POX example: Discovery Function Set Assignment

Step	Description
1	(See registration examples)
2	(See registration examples)
3	(See registration examples)
4	(See registration examples)
5	(See registration examples)
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}` `<EndDevice href="/edev/3" subscribable="0" xmlns="urn:ieee:std:2030.5:ns">` `  <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>` `  <changedTime>1379905200</changedTime>` `  <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`
11	Server responds with the FunctionSetAssignment resource. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FunctionSetAssignmentsList all="2" href="/edev/3/fsal" results="2"` `subscribable="0" xmlns="urn:ieee:std:2030.5:ns">` `  <FunctionSetAssignments href="/edev/3/fsal/0" subscribable="0">` `  <DemandResponseProgramListLink all="2" href="/edev/3/fsa1/0/drpl"/>` `  <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>`
12	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)
13	(See DemandResponse examples)

Copyright © 2018 IEEE. All rights reserved.

C.5 Discovery: Without Function Set Assignment

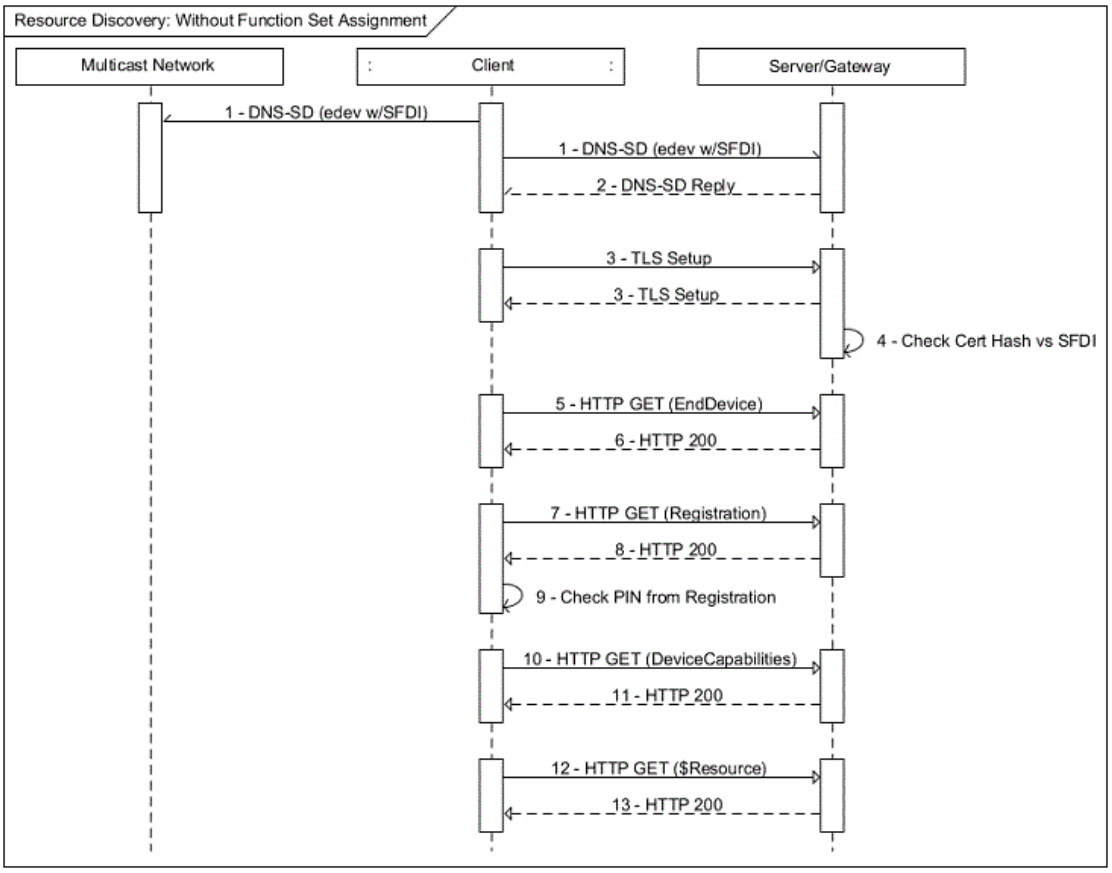


Figure C.4—Discovery without Function Set Assignment

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.5—POX example: Discovery without Function Set Assignment

Step	Description
1	(See registration examples)
2	(See registration examples)
3	(See registration examples)
4	(See registration examples)
5	(See registration examples)
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}` `<EndDevice href="/edev/3" subscribable="0" xmlns="urn:ieee:std:2030.5:ns">` `  <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>` `  <changedTime>1379905200</changedTime>` `  <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}` `<DeviceCapability href="/dcap" xmlns="urn:ieee:std:2030.5:ns">` `  <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)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.6 Discovery: Undirected without Function Set Assignment

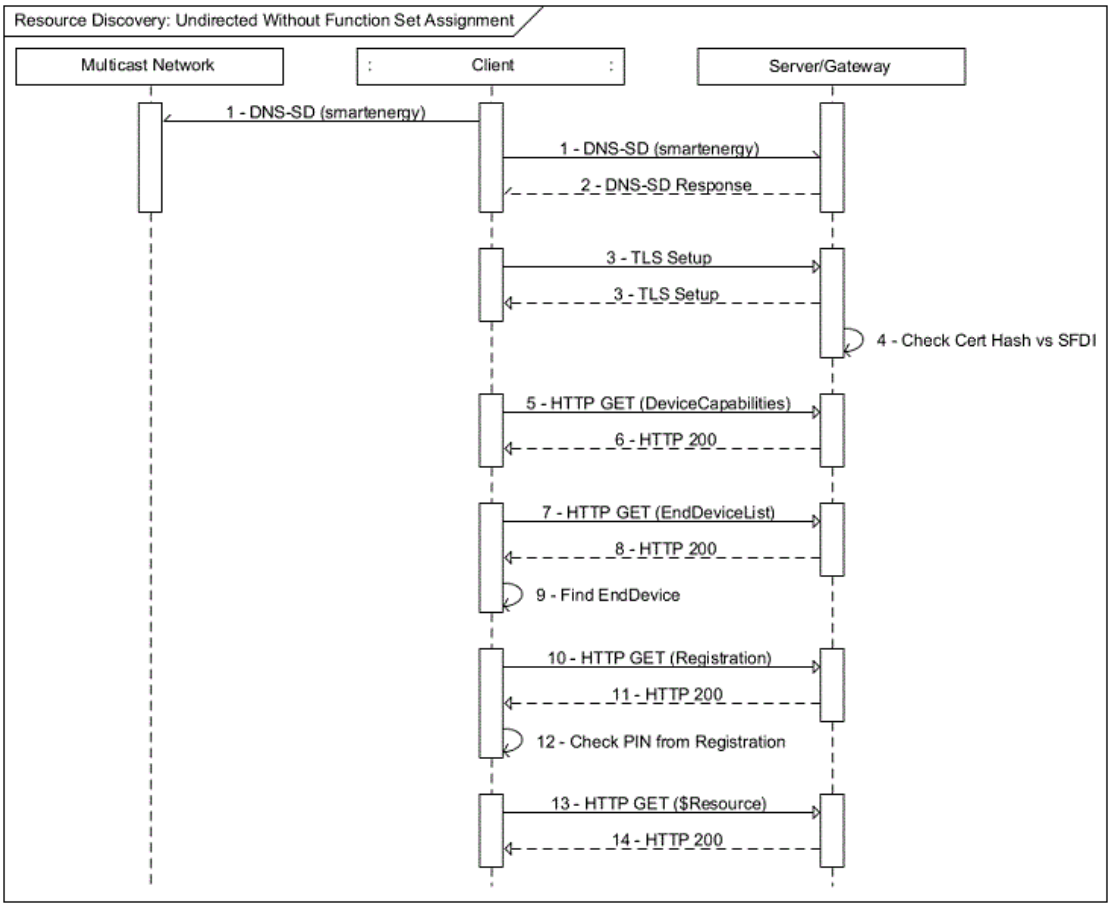


Figure C.5—Discovery undirected without Function Set Assignment

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*.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.6—POX example: Discovery undirected without Function Set Assignment

Step	Description
1	(See registration: Local example)
2	(See registration: Local example)
3	(See registration: Local example)
4	(See registration: Local example)
5	(See resource discovery: Without Function Set Assignment example)
6	(See resource discovery: Without Function Set Assignment example)
7	(See registration: Local example)
8	(See registration: Local example)
9	The client searches through the list of EndDevice records to find one with a matching SFDI. If not found, the server contains no records for the client.
10	(See registration: Remote example)
11	(See registration: Remote example)
12	(See registration: Remote example)
13	(See resource discovery: Without Function Set Assignment example)
14	(See resource discovery: Without Function Set Assignment example)

C.7 Subscription/Notification

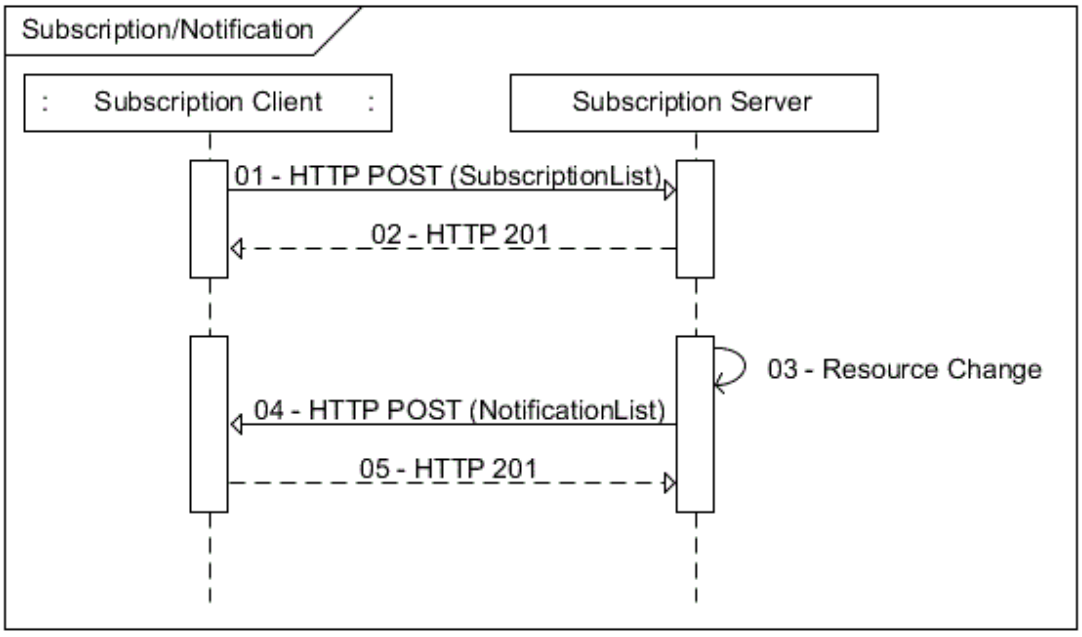


Figure C.6—Subscription/Notification

NOTE—An XML example in POX encoding is provided along with the EXI equivalent.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.7—POX example: Subscription/Notification

Step	Description
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}` `<Subscription xmlns="urn:ieee:std:2030.5:ns">` `  <subscribedResource>/upt/0/mr/4/r</subscribedResource>` `  <encoding>0</encoding>` `  <level>+S1</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}` `<Notification xmlns="urn:ieee:std:2030.5:ns"` `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">` `  <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`

Copyright © 2018 IEEE. All rights reserved.

Table C.8—EXI example: Subscription/Notification

Step	Description
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>-S1</level>* `POST /edev/8/sub HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep-exi` `Content-Length: {contentLength}` `a030 114c c716 41e5 eeae 0e85 e605 edae` `45e6 85ee 4400 014a d4cc 4008 072f 6e6f` `7465 00` (35 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}` `a030 114c c6bc 41e5 eeae 0e85 e605 edae` `45e6 85ee 4614 010a 4000 3935 e601 8274` `8390 0007 97b2 b232 bb17 9c17 b9ba b117` `9a80` (50 bytes in binary, shown in hexadecimal)
5	(Same as POX example)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.8 Demand response: General

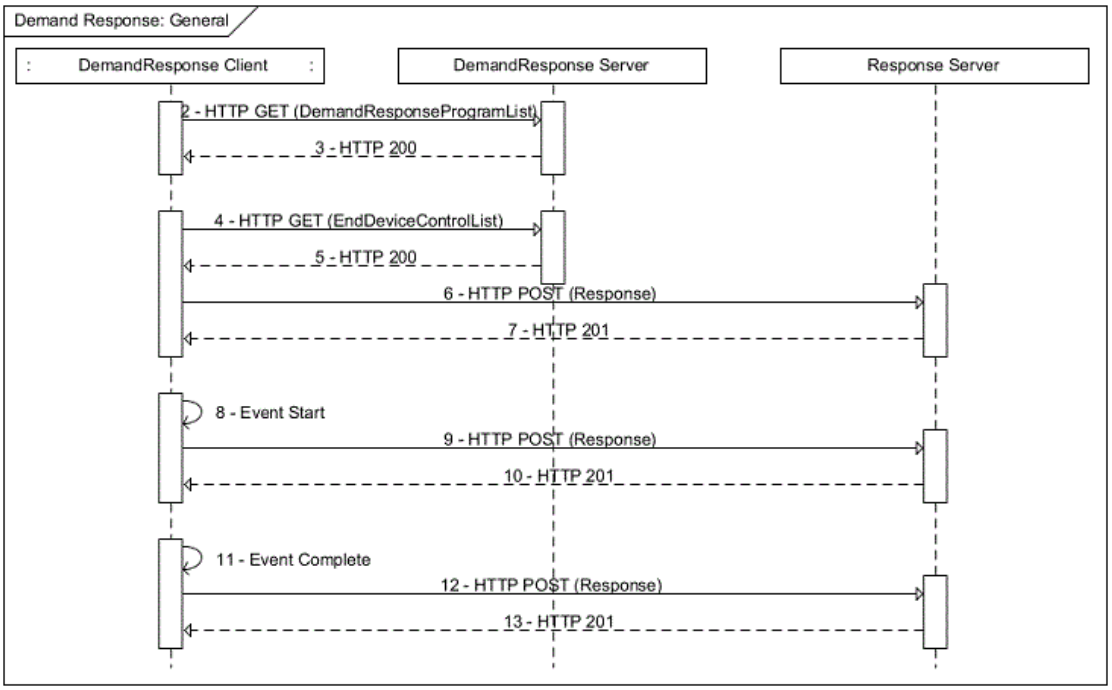


Figure C.7—Demand response: General

Table C.9—POX example: Demand response: general

Step	Description
1	The client has discovered its EndDevice instance on the server with a link to its FunctionSetAssignments. Within its FunctionSetAssignments, the client discovers it is part of a DemandResponseProgram. The enrollment is provided out of band.
2	Client GETs the list of DemandResponsePrograms from the Demand Response Load Control (DRLC) server. Client sends the following request: `GET /drp?l=2 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.9—POX example: Demand response: general *(continued)*

Step	Description
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}` `<DemandResponseProgramList all="2" results="2" subscribable="0"` `xmlns="urn:ieee:std:2030.5:ns">` `  <DemandResponseProgram href="/drp/1">` `  <mRID>0FB7</mRID>` `  <description>Operation X</description>` `  <ActiveEndDeviceControlListLink all="0" href="/drp/1/aedc"/>` `  <EndDeviceControlListLink all="1" href="/drp/1/edc"/>` `  <primacy>0</primacy>` `  </DemandResponseProgram>` `  <DemandResponseProgram href="/drp/2">` `  <mRID>80000001</mRID>` `  <description>The Wackness</description>` `  <ActiveEndDeviceControlListLink all="0" href="/drp/2/aedc"/>` `  <EndDeviceControlListLink all="0" href="/drp/2/edc"/>` `  <primacy>1</primacy>` `  </DemandResponseProgram>` `</DemandResponseProgramList>`
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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.9—POX example: Demand response: general *(continued)*

Step	Description
5	DemandResponse server responds with the list of EndDeviceControls. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<EndDeviceControlList all="1" results="1" subscribable="0"` `xmlns="urn:ieee:std:2030.5:ns">` `  <EndDeviceControl href="/drp/1/edc" replyTo="{hostname}/rsp"` `responseRequired="01" subscribable="0">` `  <mRID>CAFEFEED</mRID>` `  <description>Emergency One Hour Coffee Brew</description>` `  <creationTime>1234556</creationTime>` `  <EventStatus>` `  <currentStatus>0</currentStatus>` `  <dateTime>1234556</dateTime>` `  <potentiallySuperseded>false</potentiallySuperseded>` `  <reason>Need Caffeine Soon</reason>` `  </EventStatus>` `  <interval>` `  <duration>360</duration>` `  <start>1234900</start>` `  </interval>` `  <randomizeDuration>60</randomizeDuration>` `  <randomizeStart>60</randomizeStart>` `  <deviceCategory>08</deviceCategory>` `  <drProgramMandatory>true</drProgramMandatory>` `  <loadShiftForward>true</loadShiftForward>` `  <SetPoint>` `  <heatingSetpoint>10000</heatingSetpoint>` `  </SetPoint>` `  </EndDeviceControl>` `</EndDeviceControlList>`
6	For each EndDeviceControl with ResponseRequired Bit 0 set, the client POSTs a response with a status of “Event Received” to the response resource specified by the replyTo field in the EndDeviceControl. Client sends the following: `POST /rsp HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<DrResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1234560</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>1</status>` `  <subject>CAFEFEED</subject>` `</DrResponse>`
7	Response server replies with Status 201 Created: `HTTP/1.1 201 Created`
8	Client begins the event at the defined start time.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.9—POX example: Demand response: general *(continued)*

Step	Description
9	Client POSTs 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}` `<DrResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1234900</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>2</status>` `  <subject>CAFEFEED</subject>` `</DrResponse>`
10	Response server responds with Status 201 Created: `HTTP/1.1 201 Created`
11	Client completes the event.
12	Client POSTs 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}` `<DrResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1235260</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>3</status>` `  <subject>CAFEFEED</subject>` `</DrResponse>`
13	Response server responds with Status 201 Created: `HTTP/1.1 201 Created`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.9 Demand response: Cancel

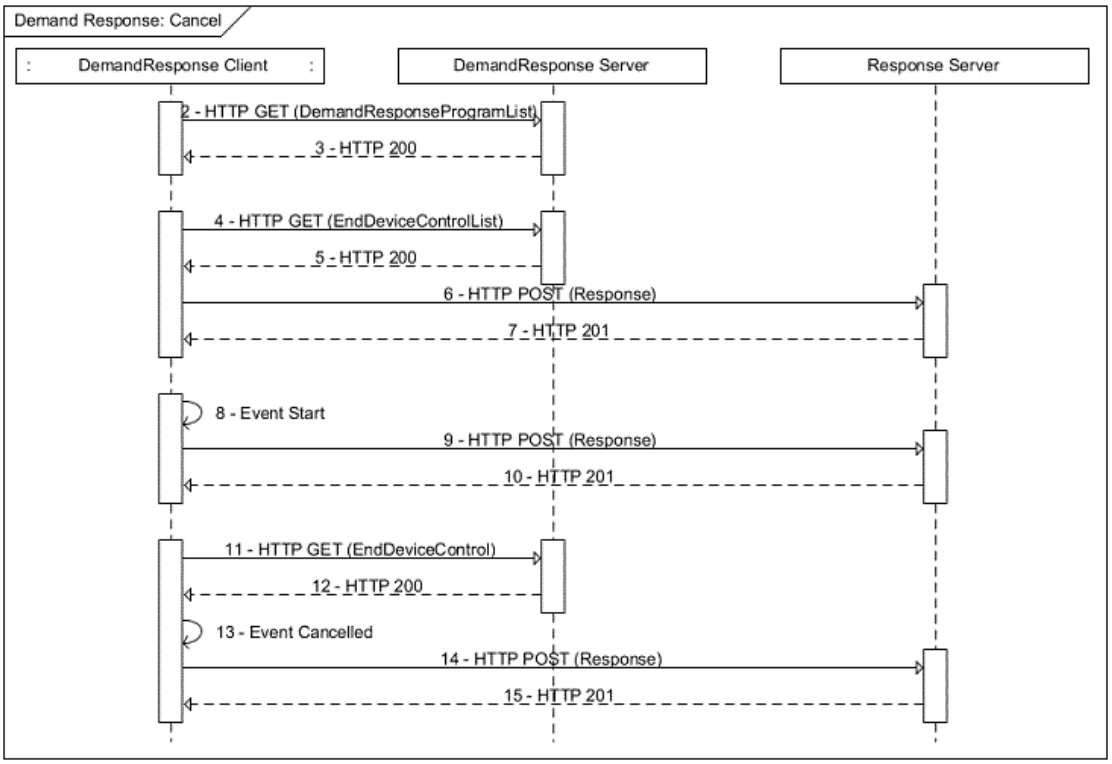


Figure C.8—Demand response: cancel

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.10—POX example: Demand response: cancel

Step	Description
1	(Same as Demand Response: General)
2	(Same as Demand Response: General)
3	(Same as Demand Response: General)
4	(Same as Demand Response: General)
5	(Same as Demand Response: General)
6	(Same as Demand Response: General)
7	(Same as Demand Response: General)
8	(Same as Demand Response: General)
9	(Same as Demand Response: General)
10	(Same as Demand Response: General)
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}` `<EndDeviceControl href="/drp/1/edc" replyTo="{hostname}/rsp" responseRequired="01" subscribable="0" xmlns="urn:ieee:std:2030.5:ns">` `  <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". Client sends the following request: `POST /rsp HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<DrResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1235100</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>6</status>` `  <subject>CAFEFEED</subject>` `</DrResponse>`
15	Response server replies with Status 201 created: `HTTP/1.1 201 Created`

Copyright © 2018 IEEE. All rights reserved.

C.10 Distributed energy resource: General

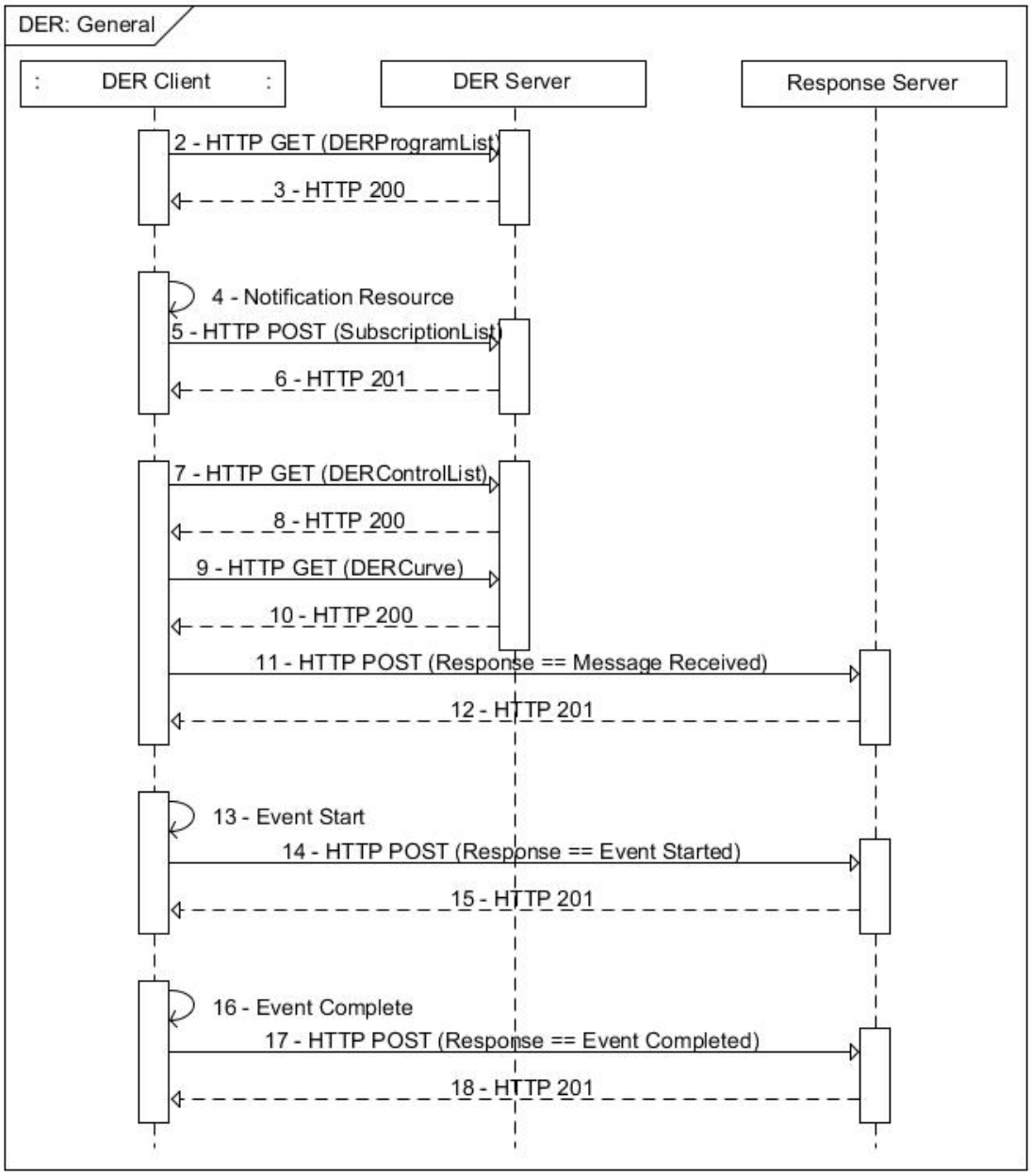


Figure C.9—Distributed energy resource: general

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.11—POX example: Distributed energy resource: general

Step	Description
1	The client discovers a server of its EndDevice instance and GETs its FunctionSetAssignments (FSA). Through its FSA the client discovers it is enrolled in a DERProgram (enrollment occurs out of band). The client registers with the specified DERProgram server if a secure connection is required.
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=+S1`
3	The DER server responds with the requested DERProgramList, for example: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<DERProgramList all="1" href="/derp" results="1" xmlns="urn:ieee:std:2030.5:ns">` `  <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="urn:ieee:std:2030.5:ns">` `  <subscribedResource>http://server.example.com/derp/0/derc</subscribedResource>` `  <encoding>0</encoding>` `  <level>+S1</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}` `Accept: application/sep+xml; level=+S1`

Copyright © 2018 IEEE. All rights reserved.

Table C.11—POX example: Distributed energy resource: general *(continued)*

Step	Description
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 all="2" href="/derp/0/derc" results="1" subscribable="1"` `xmlns="urn:ieee:std:2030.5:ns">` `  <DERControl>` `  <mRID>02BE7A7E57</mRID>` `  <description>Example DERControl 1</description>` `  <creationTime>1341446390</creationTime>` `  <EventStatus>` `  <currentStatus>1</currentStatus>` `  <dateTime>1341532800</dateTime>` `  <potentiallySuperseded>false</potentiallySuperseded>` `  </EventStatus>` `  <interval>` `  <duration>86400</duration>` `  <start>1341446400</start>` `  </interval>` `  <randomizeDuration>180</randomizeDuration>` `  <randomizeStart>180</randomizeStart>` `  <DERControlBase>` `  <opModVoltWatt 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=+S1`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.11—POX example: Distributed energy resource: general *(continued)*

Step	Description
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 4) 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 href="/derp/0/dc/3" xmlns="urn:ieee:std:2030.5:ns">` `  <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>101</xvalue>` `  <yvalue>-50</yvalue>` `  </CurveData>` `  <CurveData>` `  <xvalue>97</xvalue>` `  <yvalue>50</yvalue>` `  </CurveData>` `  <curveType>11</curveType>` `  <rampDecTms>600</rampDecTms>` `  <rampIncTms>600</rampIncTms>` `  <rampPT1Tms>10</rampPT1Tms>` `  <xMultiplier>0</xMultiplier>` `  <yMultiplier>0</yMultiplier>` `  <yRefType>3</yRefType>` `</DERCurve>`
11	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: `POST /rsp HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<DERControlResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1341507000</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>1</status>` `  <subject>02BE7A7E57</subject>` `</DERControlResponse>`
12	Response server responds with: `HTTP/1.1 201 Created`
13	Client begins the event at the specified start (or current) time.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.11—POX example: Distributed energy resource: general *(continued)*

Step	Description
14	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: `POST /rsp HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<DERControlResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1341507010</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>2</status>` `  <subject>02BE7A7E57</subject>` `</DERControlResponse>`
15	Response server responds with: `HTTP/1.1 201 Created`
16	Client completes the event.
17	Client POSTs a response with a status of "Event Completed" to the response resource specified by the replyTo field in the DERControl. Client sends the following: `POST /rsp HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<DERControlResponse xmlns="urn:ieee:std:2030.5:ns">` `  <createdDateTime>1341532810</createdDateTime>` `  <endDeviceLFDI>C0FFEE00</endDeviceLFDI>` `  <status>3</status>` `  <subject>02BE7A7E57</subject>` `</DERControlResponse>`
18	Response server responds with: `HTTP/1.1 201 Created`

C.11 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 four TOU tiers and no blocks and that we want to read the current tier three consumption.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

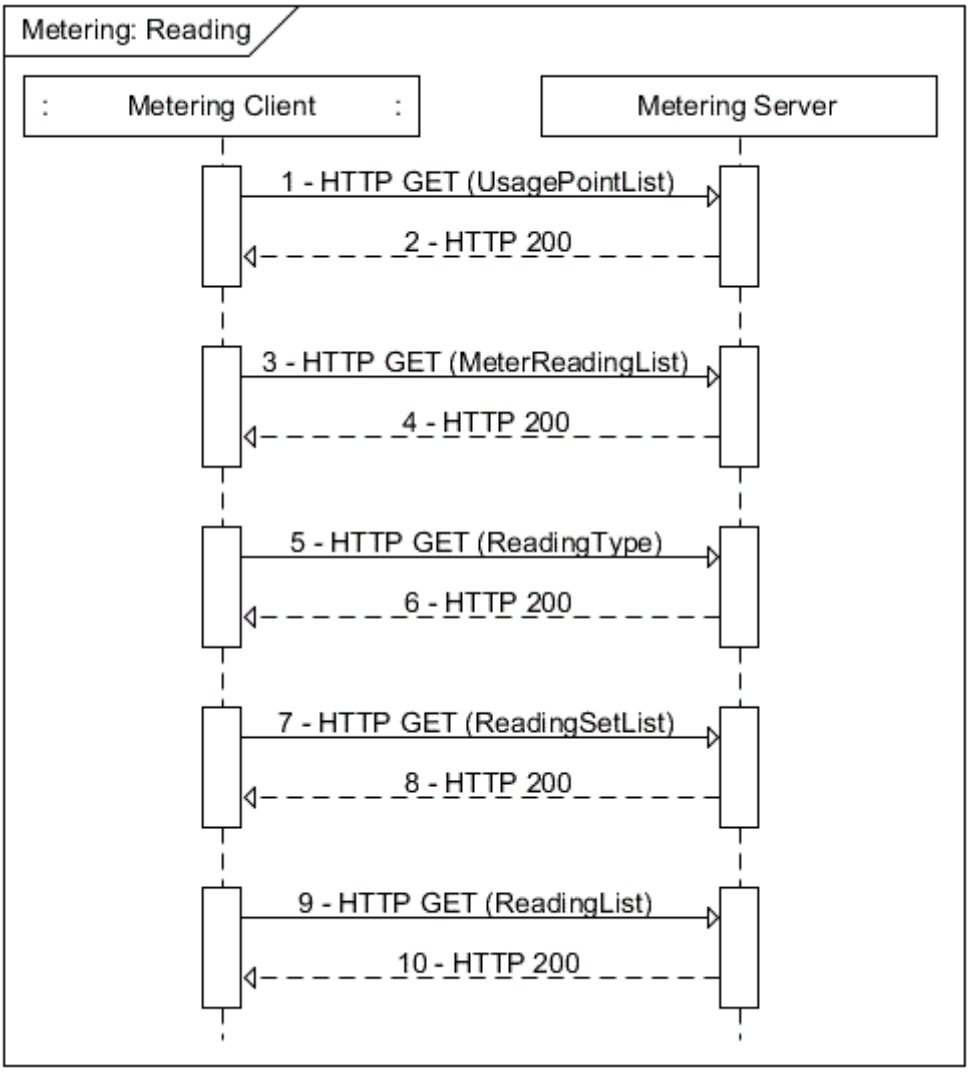


Figure C.10—Meter reading

NOTE—In most cases, registration is required to obtain access to metering.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.12—POX example: Meter reading

Step	Description
1	Client GETs the UsagePointList from the Metering server. Note: If directed through FunctionSetAssignments 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`
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="urn:ieee:std:2030.5:ns">` `  <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 three steps may be repeated for each page required to read the entire list. For this example, we are requesting up to ten MeterReadings at a time. Subsequent GETs would increment the “s” query parameter by ten, 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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.12—POX example: Meter reading *(continued)*

Step	Description
4	Metering server replies with up to ten MeterReadingList instances. Only six 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="urn:ieee:std:2030.5:ns">` `  <MeterReading href="/upt/0/mr/5">` `  <mRID>1500006CC8</mRID>` `  <description>Instantaneous Reading for VAR's</description>` `  <ReadingLink href="/upt/0/mr/5/r"/>` `  <ReadingTypeLink href="/upt/0/mr/5/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/0/mr/4">` `  <mRID>1400006CC8</mRID>` `  <description>Instantaneous Reading for Wh</description>` `  <ReadingLink href="/upt/0/mr/4/r"/>` `  <ReadingTypeLink href="/upt/0/mr/4/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/0/mr/3">` `  <mRID>1200006CC8</mRID>` `  <description>Interval Reading for VAR's</description>` `  <ReadingSetListLink all="24" href="/upt/0/mr/3/rs"/>` `  <ReadingTypeLink href="/upt/0/mr/3/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/1">` `  <mRID>0E00006CC8</mRID>` `  <description>Cumulative Reading for VAR's</description>` `  <ReadingLink href="/upt/0/mr/1/r"/>` `  <ReadingSetListLink all="1" href="/upt/0/mr/1/rs"/>` `  <ReadingTypeLink href="/upt/0/mr/1/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/0/mr/0">` `  <mRID>0C00006CC8</mRID>` `  <description>Cumulative Reading for Wh</description>` `  <ReadingLink href="/upt/0/mr/0/r"/>` `  <ReadingSetListLink all="1" href="/upt/0/mr/0/rs"/>` `  <ReadingTypeLink href="/upt/0/mr/0/rt"/>` `  </MeterReading>` `</MeterReadingList>`
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 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.12—POX example: Meter reading *(continued)*

Step	Description
6	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/0/mr/0/rt" xmlns="urn:ieee:std:2030.5:ns">` `  <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 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/0/mr/0/rs?s=0&l=4 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`
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="1" href="/upt/0/mr/0/rs" results="1" subscribable="0"` `xmlns="urn:ieee:std:2030.5:ns">` `  <ReadingSet href="/upt/0/mr/0/rs/0">` `  <mRID>0D00006CC8</mRID>` `  <description>Cumulative Reading Set for Whrs</description>` `  <timePeriod>` `  <duration>7737</duration>` `  <start>1338817886</start>` `  </timePeriod>` `  <ReadingListLink all="5" href="/upt/0/mr/0/rs/0/r"/>` `  </ReadingSet>` `</ReadingSetList>`
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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.12—POX example: Meter reading *(continued)*

Step	Description
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="5" href="/upt/0/mr/0/rs/0/r" results="1"` `xmlns="urn:ieee:std:2030.5:ns">` `  <Reading href="/upt/0/mr/0/rs/0/r/3">` `  <value>361231</value>` `  </Reading>` `</ReadingList>`

C.12 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

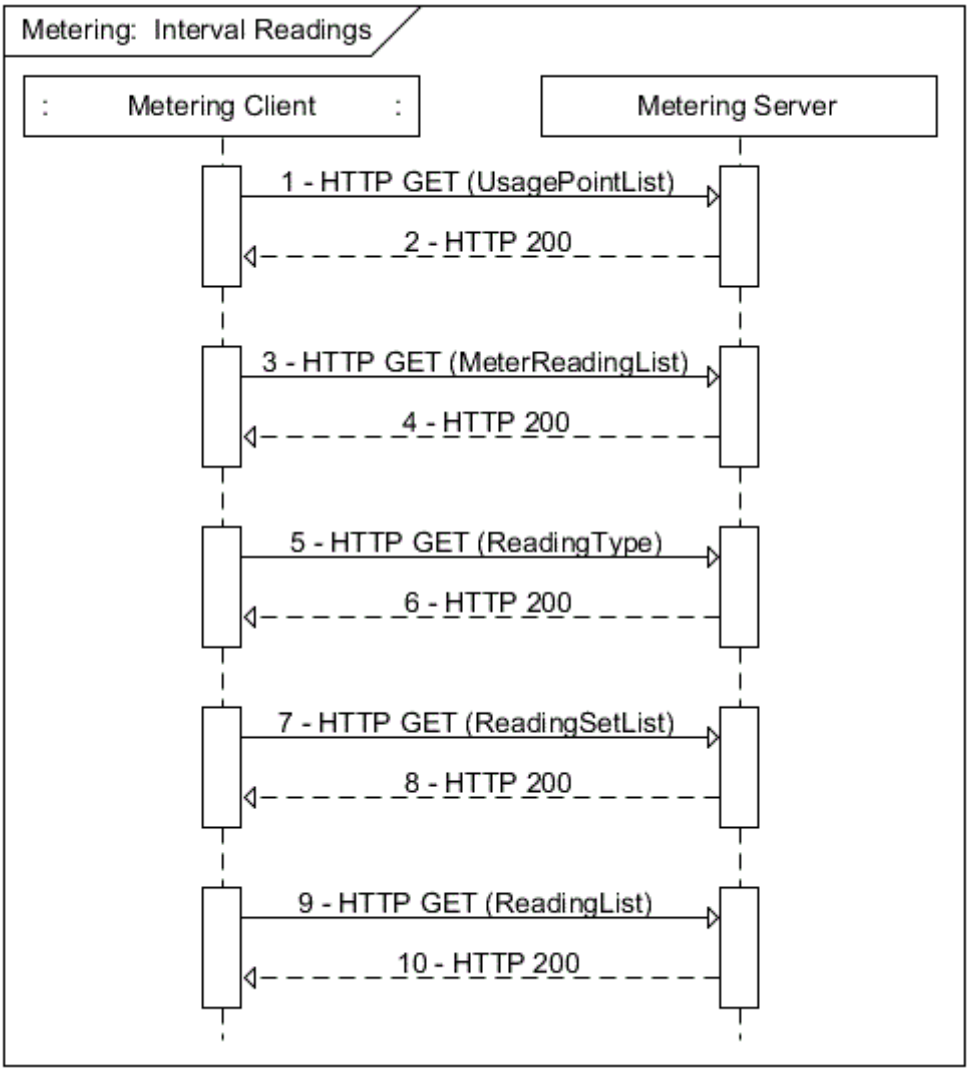


Figure C.11—Metering interval

NOTE—In most cases, registration is required to obtain access to metering.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.13—POX example: Metering interval

Step	Description
1	Client GETs the UsagePointList from the Metering server. Note: If directed through FunctionSetAssignments 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`
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="urn:ieee:std:2030.5:ns">` `  <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 three steps may be repeated for each page required to read the entire list. For this example, we are requesting up to ten MeterReadings at a time. Subsequent GETs would increment the “s” query parameter by ten 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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.13—POX example: Metering interval *(continued)*

Step	Description
4	Metering server replies with up to ten MeterReadingList instances. Only six 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="urn:ieee:std:2030.5:ns">` `  <MeterReading href="/upt/0/mr/5">` `  <mRID>1500006CC8</mRID>` `  <description>Instantaneous Reading for VAR’s</description>` `  <ReadingLink href="/upt/0/mr/5/r"/>` `  <ReadingTypeLink href="/upt/0/mr/5/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/0/mr/4">` `  <mRID>1400006CC8</mRID>` `  <description>Instantaneous Reading for Wh</description>` `  <ReadingLink href="/upt/0/mr/4/r"/>` `  <ReadingTypeLink href="/upt/0/mr/4/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/0/mr/3">` `  <mRID>1200006CC8</mRID>` `  <description>Interval Reading for VAR’s</description>` `  <ReadingSetListLink all="24" href="/upt/0/mr/3/rs"/>` `  <ReadingTypeLink href="/upt/0/mr/3/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/1">` `  <mRID>0E00006CC8</mRID>` `  <description>Cumulative Reading for VAR’s</description>` `  <ReadingLink href="/upt/0/mr/1/r"/>` `  <ReadingSetListLink all="1" href="/upt/0/mr/1/rs"/>` `  <ReadingTypeLink href="/upt/0/mr/1/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/0/mr/0">` `  <mRID>0C00006CC8</mRID>` `  <description>Cumulative Reading for Wh</description>` `  <ReadingLink href="/upt/0/mr/0/r"/>` `  <ReadingSetListLink all="1" href="/upt/0/mr/0/rs"/>` `  <ReadingTypeLink href="/upt/0/mr/0/rt"/>` `  </MeterReading>` `</MeterReadingList>`
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?s=0&l=1 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.13—POX example: Metering interval *(continued)*

Step	Description
6	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/0/mr/0/rt" xmlns="urn:ieee:std:2030.5:ns">` `  <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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.13—POX example: Metering interval *(continued)*

Step	Description
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="urn:ieee:std:2030.5:ns">` `  <ReadingSet href="/upt/0/mr/2/rs/2">` `  <mRID>2000006CC8</mRID>` `  <description>Reading Set for WHrs</description>` `  <timePeriod>` `  <duration>3600</duration>` `  <start>1338842400</start>` `  </timePeriod>` `  <ReadingListLink all="12" href="/upt/0/mr/2/rs/2/r"/>` `  </ReadingSet>` `  <ReadingSet href="/upt/0/mr/2/rs/4">` `  <mRID>2200006CC8</mRID>` `  <description>Reading Set for WHrs</description>` `  <timePeriod>` `  <duration>3600</duration>` `  <start>1338846000</start>` `  </timePeriod>` `  <ReadingListLink all="12" href="/upt/0/mr/2/rs/4/r"/>` `  </ReadingSet>` `  <ReadingSet href="/upt/0/mr/2/rs/6">` `  <mRID>2400006CC8</mRID>` `  <description>Reading Set for WHrs</description>` `  <timePeriod>` `  <duration>3600</duration>` `  <start>1338849600</start>` `  </timePeriod>` `  <ReadingListLink all="12" href="/upt/0/mr/2/rs/6/r"/>` `  </ReadingSet>` `  <ReadingSet href="/upt/0/mr/2/rs/8">` `  <mRID>2600006CC8</mRID>` `  <description>Reading Set for WHrs</description>` `  <timePeriod>` `  <duration>3600</duration>` `  <start>1338853200</start>` `  </timePeriod>` `  <ReadingListLink all="12" href="/upt/0/mr/2/rs/8/r"/>` `  </ReadingSet>` `</ReadingSetList>`
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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.13—POX example: Metering interval *(continued)*

Step	Description
10	Metering server replies with ReadingList. 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="urn:ieee:std:2030.5:ns">` `  <Reading href="/upt/0/mr/2/rs/4/r/0">` `  <value>1163</value>` `  <localID>00</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/2">` `  <value>1162</value>` `  <localID>01</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/4">` `  <value>1163</value>` `  <localID>02</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/6">` `  <value>1163</value>` `  <localID>03</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/8">` `  <value>1163</value>` `  <localID>04</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/a">` `  <value>1163</value>` `  <localID>05</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/c">` `  <value>1162</value>` `  <localID>06</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/e">` `  <value>1163</value>` `  <localID>07</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/10">` `  <value>1163</value>` `  <localID>08</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/12">` `  <value>1163</value>` `  <localID>09</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/14">` `  <value>1162</value>` `  <localID>0A</localID>` `  </Reading>` `  <Reading href="/upt/0/mr/2/rs/4/r/16">` `  <value>1163</value>` `  <localID>0B</localID>` `  </Reading>` `</ReadingList>` 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.13 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.

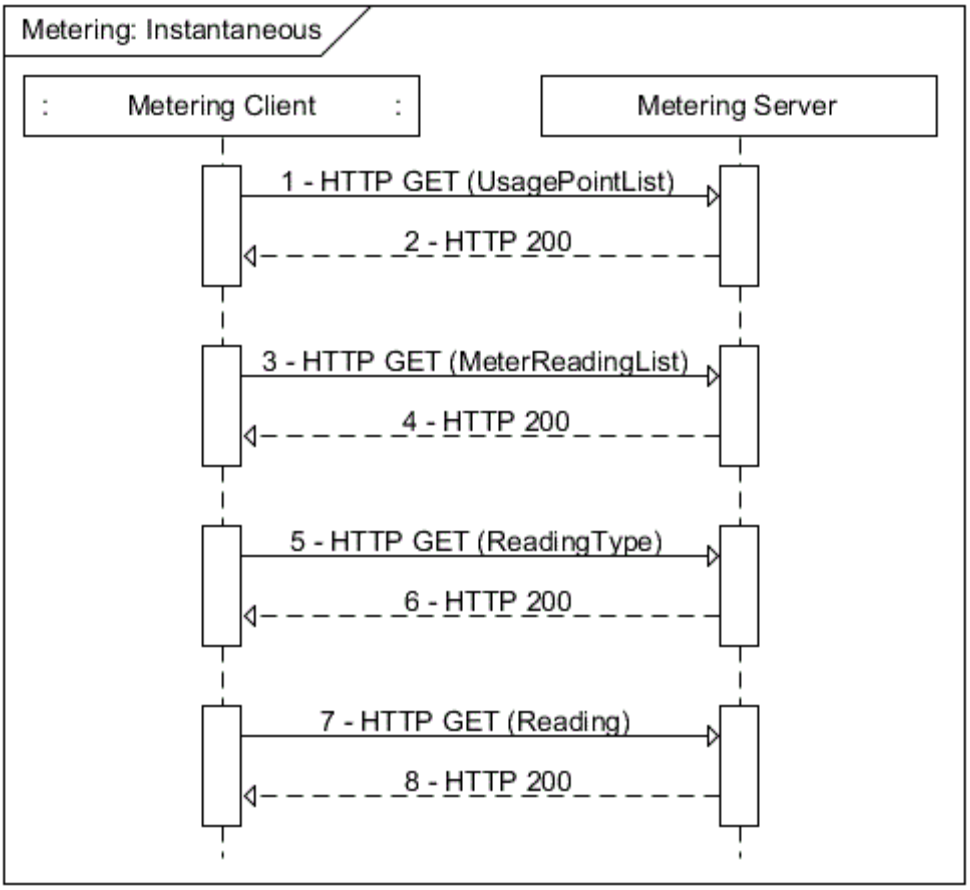


Figure C.12—Metering instantaneous

NOTE—In most cases, registration is required to obtain access to metering.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.14—POX example: Metering instantaneous

Step	Description
1	Client GETs the UsagePointList from the Metering server. Note: If directed through FunctionSetAssignments 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`
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="urn:ieee:std:2030.5:ns">` `  <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 three steps may be repeated for each page required to read the entire list. For this example, we are requesting up to ten MeterReadings at a time. Subsequent GETs would increment the “s” query parameter by ten 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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.14—POX example: Metering instantaneous *(continued)*

Step	Description
4	Metering server replies with up to ten MeterReading instances. Only six 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="urn:ieee:std:2030.5:ns">` `<MeterReading href="/upt/0/mr/5">` `<mRID>1500006CC8</mRID>` `<description>Instantaneous Reading for VAR's</description>` `<ReadingLink href="/upt/0/mr/5/r"/>` `<ReadingTypeLink href="/upt/0/mr/5/rt"/>` `</MeterReading>` `<MeterReading href="/upt/0/mr/4">` `<mRID>1400006CC8</mRID>` `<description>Instantaneous Reading for Wh</description>` `<ReadingLink href="/upt/0/mr/4/r"/>` `<ReadingTypeLink href="/upt/0/mr/4/rt"/>` `</MeterReading>` `<MeterReading href="/upt/0/mr/3">` `<mRID>1200006CC8</mRID>` `<description>Interval Reading for VAR's</description>` `<ReadingSetListLink all="24" href="/upt/0/mr/3/rs"/>` `<ReadingTypeLink href="/upt/0/mr/3/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/1">` `<mRID>0E00006CC8</mRID>` `<description>Cumulative Reading for VAR's</description>` `<ReadingLink href="/upt/0/mr/1/r"/>` `<ReadingSetListLink all="1" href="/upt/0/mr/1/rs"/>` `<ReadingTypeLink href="/upt/0/mr/1/rt"/>` `</MeterReading>` `<MeterReading href="/upt/0/mr/0">` `<mRID>0C00006CC8</mRID>` `<description>Cumulative Reading for Wh</description>` `<ReadingLink href="/upt/0/mr/0/r"/>` `<ReadingSetListLink all="1" href="/upt/0/mr/0/rs"/>` `<ReadingTypeLink href="/upt/0/mr/0/rt"/>` `</MeterReading>` `</MeterReadingList>`
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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.14—POX example: Metering instantaneous *(continued)*

Step	Description
6	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/0/mr/4/rt" xmlns="urn:ieee:std:2030.5:ns">` `  <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`
8	Metering server replies with the Reading. A typical response looks like: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<Reading href="/upt/0/mr/4/r" xmlns="urn:ieee:std:2030.5:ns">` `  <value>14</value>` `</Reading>` 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.

C.14 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

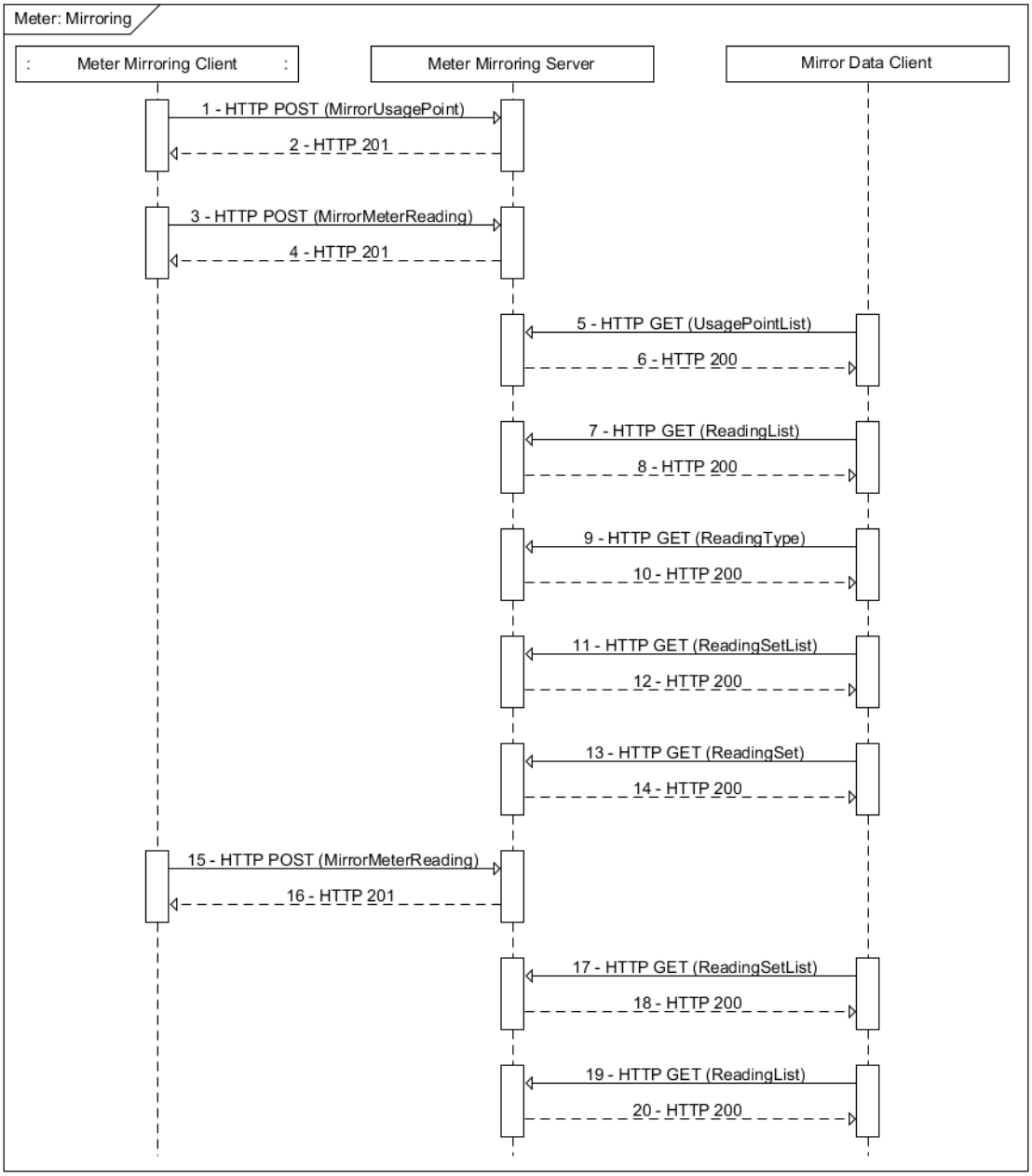


Figure C.13—Meter Mirroring

NOTE—In most cases, registration is required to obtain access to Metering and Meter Mirroring.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.15—POX example: Meter Mirroring

Step	Description
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="urn:ieee:std:2030.5:ns">` `  <mRID>0600006CC8</mRID>` `  <description>Gas Mirroring</description>` `  <roleFlags>13</roleFlags>` `  <serviceCategoryKind>1</serviceCategoryKind>` `  <status>1</status>` `  <deviceLFDI>00</deviceLFDI>` `  <MirrorMeterReading>` `  <mRID>0700006CC8</mRID>` `  <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>` `</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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.15—POX example: Meter Mirroring *(continued)*

Step	Description
3	Meter Mirroring client POSTs a MirrorMeterReading to the Mirror Metering server. Client sends the following request:

```
POST /mup/0 HTTP/1.1
Host: {hostname}
Content-Type: application/sep+xml
Content-Length: {contentLength}

<MirrorMeterReading xmlns="urn:ieee:std:2030.5:ns">
    <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>
        <Reading>
            <value>8</value>
            <localID>0C</localID>
        </Reading>
        <Reading>
            <value>9</value>
            <localID>0D</localID>
        </Reading>
        <Reading>
            <value>10</value>
            <localID>0E</localID>
        </Reading>
        <Reading>
```


Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

<table>
<tr><td></td><td>

```
            <value>12</value>
            <localID>0F</localID>
        </Reading>
        <Reading>
            <value>14</value>
            <localID>10</localID>
        </Reading>
        <Reading>
            <value>13</value>
            <localID>11</localID>
        </Reading>
        <Reading>
            <value>11</value>
            <localID>12</localID>
        </Reading>
        <Reading>
            <value>7</value>
            <localID>13</localID>
        </Reading>
        <Reading>
            <value>8</value>
            <localID>14</localID>
        </Reading>
        <Reading>
            <value>10</value>
            <localID>15</localID>
        </Reading>
        <Reading>
            <value>10</value>
            <localID>16</localID>
        </Reading>
        <Reading>
            <value>10</value>
            <localID>17</localID>
        </Reading>
    </MirrorReadingSet>
</MirrorMeterReading>
```

</td></tr>
<tr><td>4</td><td>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 through 4 could be combined into two steps by including the initial interval reading set data in the initial MirrorUsagePoint POST.</td></tr>
<tr><td>5</td><td>Mirror data client GETs the UsagePointList from the Metering server.

Note: If directed through FunctionSetAssignments 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
```

</td></tr>
</table>

Copyright © 2018 IEEE. All rights reserved.

Table C.15—POX example: Meter Mirroring *(continued)*

Step	Description
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="urn:ieee:std:2030.5:ns">` `  <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 three steps may be repeated for each page required to read the entire list. For this example, we are requesting up to ten MeterReadings at a time. Subsequent GETs would increment the “s” query parameter by ten 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 two 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="urn:ieee:std:2030.5:ns">` `  <MeterReading href="/upt/1/mr/0">` `  <mRID>0700006CC8</mRID>` `  <description>Cumulative Reading for Gas</description>` `  <ReadingLink href="/upt/1/mr/0/r"/>` `  <ReadingTypeLink href="/upt/1/mr/0/rt"/>` `  </MeterReading>` `  <MeterReading href="/upt/1/mr/1">` `  <mRID>0800006CC8</mRID>` `  <description> Interval Readings for Gas</description>` `  <ReadingSetListLink all="1" href="/upt/1/mr/1/rs"/>` `  <ReadingTypeLink href="/upt/1/mr/1/rt"/>` `  </MeterReading>` `</MeterReadingList>`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.15—POX example: Meter Mirroring *(continued)*

Step	Description
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`
10	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="urn:ieee:std:2030.5:ns">` `  <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.
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`
12	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="urn:ieee:std:2030.5:ns">` `  <ReadingSet href="/upt/1/mr/1/rs/32">` `  <mRID>2000006CC8</mRID>` `  <timePeriod>` `  <duration>86400</duration>` `  <start>1341579365</start>` `  </timePeriod>` `  <ReadingListLink all="24" href="/upt/1/mr/1/rs/32/r"/>` `  </ReadingSet>` `</ReadingSetList>`
13	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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.15—POX example: Meter Mirroring *(continued)*

Step	Description
14	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"` `xmlns="urn:ieee:std:2030.5:ns">` `<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>` `<localID>0A</localID>` `</Reading>` `<Reading href="/upt/1/mr/1/rs/32/r/f">` `<value>5</value>` `<localID>0B</localID>` `</Reading>` `<Reading href="/upt/1/mr/1/rs/32/r/10">` `<value>8</value>` `<localID>0C</localID>` `</Reading>` `<Reading href="/upt/1/mr/1/rs/32/r/11">` `<value>9</value>` `<localID>0D</localID>` `</Reading>` `<Reading href="/upt/1/mr/1/rs/32/r/12">` `<value>10</value>` `<localID>0E</localID>` `</Reading>` `<Reading href="/upt/1/mr/1/rs/32/r/13">` `<value>12</value>` `<localID>0F</localID>` `</Reading>` `<Reading href="/upt/1/mr/1/rs/32/r/14">` `<value>14</value>` `<localID>10</localID>` `</Reading>`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

```
    <Reading href="/upt/1/mr/1/rs/32/r/15">
        <value>13</value>
        <localID>11</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/32/r/16">
        <value>11</value>
        <localID>12</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/32/r/17">
        <value>7</value>
        <localID>13</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/32/r/0">
        <value>8</value>
        <localID>14</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/32/r/1">
        <value>10</value>
        <localID>15</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/32/r/2">
        <value>10</value>
        <localID>16</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/32/r/3">
        <value>10</value>
        <localID>17</localID>
    </Reading>
</ReadingList>
```

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}
Content-Type: application/sep+xml
Content-Length: {contentLength}

<MirrorMeterReadingList xmlns="urn:ieee:std:2030.5:ns">
    <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>
```


Copyright © 2018 IEEE. All rights reserved.

```
                <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>
            <Reading>
                <value>8</value>
                <localID>0C</localID>
            </Reading>
            <Reading>
                <value>9</value>
                <localID>0D</localID>
            </Reading>
            <Reading>
                <value>10</value>
                <localID>0E</localID>
            </Reading>
            <Reading>
                <value>12</value>
                <localID>0F</localID>
            </Reading>
            <Reading>
                <value>14</value>
                <localID>10</localID>
            </Reading>
            <Reading>
                <value>13</value>
                <localID>11</localID>
            </Reading>
            <Reading>
                <value>11</value>
                <localID>12</localID>
            </Reading>
            <Reading>
                <value>7</value>
                <localID>13</localID>
            </Reading>
            <Reading>
                <value>8</value>
                <localID>14</localID>
            </Reading>
            <Reading>
                <value>10</value>
                <localID>15</localID>
            </Reading>
            <Reading>
                <value>10</value>
                <localID>16</localID>
            </Reading>
            <Reading>
                <value>10</value>
                <localID>17</localID>
            </Reading>
        </MirrorReadingSet>
    </MirrorMeterReading>
</MirrorMeterReadingList>
```


Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.15—POX example: Meter Mirroring *(continued)*

Step	Description
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`
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="urn:ieee:std:2030.5:ns">` `  <ReadingSet href="/upt/1/mr/1/rs/33">` `  <mRID>A000006CC8</mRID>` `  <timePeriod>` `  <duration>86400</duration>` `  <start>1341665765</start>` `  </timePeriod>` `  <ReadingListLink all="24" href="/upt/1/mr/1/rs/33/r"/>` `  </ReadingSet>` `</ReadingSetList>`
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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.15—POX example: Meter Mirroring *(continued)*

Step	Description
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="urn:ieee:std:2030.5:ns"> <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> </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>

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

```
    </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 href="/upt/1/mr/1/rs/33/r/1">
        <value>10</value>
        <localID>15</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/33/r/2">
        <value>10</value>
        <localID>16</localID>
    </Reading>
    <Reading href="/upt/1/mr/1/rs/33/r/3">
        <value>10</value>
        <localID>17</localID>
    </Reading>
</ReadingList>
```

C.15 Pricing: Time of use

The flow diagram in Figure C.14 describes Pricing client device obtaining Pricing information.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

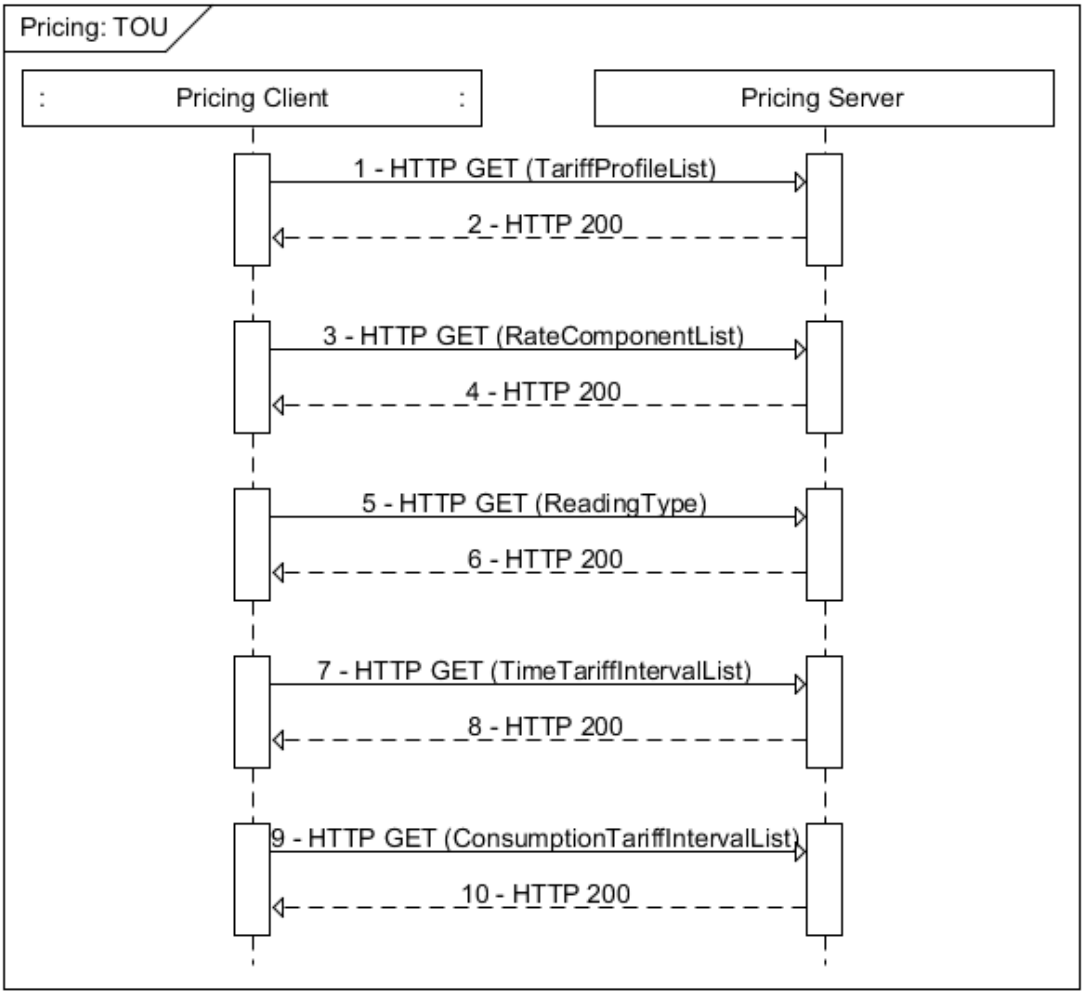


Figure C.14—Pricing time of use

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.16—POX example: Pricing TOU

Step	Description
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`
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="urn:ieee:std:2030.5:ns">` `  <mRID>799794f4620b17e00000e566</mRID>` `  <description>PEV TOU Rate</description>` `  <currency>840</currency>` `  <pricePowerOfTenMultiplier>-6</pricePowerOfTenMultiplier>` `  <primacy>0</primacy>` `  <rateCode>TOU-D-PEV Baseline 6</rateCode>` `  <RateComponentListLink all="1" href="/tp/3/rc"/>` `  <serviceCategoryKind>0</serviceCategoryKind>` `</TariffProfile>`
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" results="1"` `xmlns="urn:ieee:std:2030.5:ns">` `  <RateComponent href="/tp/3/rc/3">` `  <mRID>fc000b07143d24fc0000e566</mRID>` `  <description>TOU-D-PEV</description>` `  <ActiveTimeTariffIntervalListLink all="0" href="/tp/3/rc/3/acttti"/>` `  <flowRateEndLimit>` `  <multiplier>0</multiplier>` `  <unit>38</unit>` `  <value>400</value>` `  </flowRateEndLimit>` `  <flowRateStartLimit>` `  <multiplier>0</multiplier>` `  <unit>38</unit>` `  <value>0</value>` `  </flowRateStartLimit>` `  <ReadingTypeLink href="/rt/1"/>` `  <roleFlags>12</roleFlags>` `  <TimeTariffIntervalListLink all="5" href="/tp/3/rc/3/tti"/>` `  </RateComponent>` `</RateComponentList>`
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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.16—POX example: Pricing TOU *(continued)*

Step	Description
6	Pricing 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/1" xmlns="urn:ieee:std:2030.5:ns">` `  <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>`
7	Client GETs the TimeTariffIntervalList from the Pricing server. Client sends the following request: `GET /tp/3/rc/3/tti?l=5 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`
8	Pricing server responds with the TimeTariffIntervalList. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<TimeTariffIntervalList all="5" href="/tp/3/rc/3/tti" results="5" subscribable="1"` `xmlns="urn:ieee:std:2030.5:ns">` `  <TimeTariffInterval href="/tp/3/rc/3/tti/5" subscribable="1">` `  <mRID>ef06fa23dc0a0f650000e566</mRID>` `  <description>Off-Peak 1</description>` `  <creationTime>1357430400</creationTime>` `  <EventStatus>` `  <currentStatus>0</currentStatus>` `  <dateTime>1357430400</dateTime>` `  <potentiallySuperseded>false</potentiallySuperseded>` `  </EventStatus>` `  <interval>` `  <duration>28800</duration>` `  <start>1357516800</start>` `  </interval>` `  <randomizeDuration>300</randomizeDuration>` `  <randomizeStart>300</randomizeStart>` `  <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/5/cti"/>` `  <touTier>1</touTier>` `  </TimeTariffInterval>` `  <TimeTariffInterval href="/tp/3/rc/3/tti/6" subscribable="1">` `  <mRID>41fc7c07e16820770000e566</mRID>` `  <description>Mid-Peak 1</description>` `  <creationTime>1357430400</creationTime>` `  <EventStatus>` `  <currentStatus>0</currentStatus>` `  <dateTime>1357430400</dateTime>` `  <potentiallySuperseded>false</potentiallySuperseded>` `  </EventStatus>` `  <interval>` `  <duration>14400</duration>` `  <start>1357545600</start>` `  </interval>` `  <randomizeDuration>300</randomizeDuration>` `  <randomizeStart>300</randomizeStart>`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

<table>
<tr><td></td><td>
<pre>
        <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/6/cti"/>
        <touTier>2</touTier>
    </TimeTariffInterval>
    <TimeTariffInterval href="/tp/3/rc/3/tti/7" subscribable="1">
        <mRID>63eed7b30c1c87a40000e566</mRID>
        <description>On-Peak</description>
        <creationTime>1357430400</creationTime>
        <EventStatus>
            <currentStatus>0</currentStatus>
            <dateTime>1357430400</dateTime>
            <potentiallySuperseded>false</potentiallySuperseded>
        </EventStatus>
        <interval>
            <duration>21600</duration>
            <start>1357552800</start>
        </interval>
        <randomizeDuration>300</randomizeDuration>
        <randomizeStart>300</randomizeStart>
        <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/7/cti"/>
        <touTier>3</touTier>
    </TimeTariffInterval>
    <TimeTariffInterval href="/tp/3/rc/3/tti/8" subscribable="1">
        <mRID>9b04f0713e9212d90000e566</mRID>
        <description>Mid-Peak 2</description>
        <creationTime>1357430400</creationTime>
        <EventStatus>
            <currentStatus>0</currentStatus>
            <dateTime>1357430400</dateTime>
            <potentiallySuperseded>false</potentiallySuperseded>
        </EventStatus>
        <interval>
            <duration>18000</duration>
            <start>1357574400</start>
        </interval>
        <randomizeDuration>300</randomizeDuration>
        <randomizeStart>300</randomizeStart>
        <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/8/cti"/>
        <touTier>2</touTier>
    </TimeTariffInterval>
    <TimeTariffInterval href="/tp/3/rc/3/tti/9" subscribable="1">
        <mRID>c13c8755dc39b5950000e566</mRID>
        <description>Off-Peak 2</description>
        <creationTime>1357430400</creationTime>
        <EventStatus>
            <currentStatus>0</currentStatus>
            <dateTime>1357430400</dateTime>
            <potentiallySuperseded>false</potentiallySuperseded>
        </EventStatus>
        <interval>
            <duration>10800</duration>
            <start>1357592400</start>
        </interval>
        <randomizeDuration>300</randomizeDuration>
        <randomizeStart>300</randomizeStart>
        <ConsumptionTariffIntervalListLink all="1" href="/tp/3/rc/3/tti/9/cti"/>
        <touTier>1</touTier>
    </TimeTariffInterval>
</TimeTariffIntervalList>
</pre>
</td></tr>
<tr><td>9</td><td>Client GETs a ConsumptionTariffIntervalList from the Pricing server (note that there is a ConsumptionTariffIntervalList for each TimeTariffInterval, but only one is shown below).

Client sends the following request:

<pre>
GET /tp/3/rc/3/tti/5/cti?l=1 HTTP/1.1
Host: {hostname}
Accept: application/sep+xml
</pre>
</td></tr>
</table>

Copyright © 2018 IEEE. All rights reserved.

Table C.16—POX example: Pricing TOU *(continued)*

Step	Description
10	Pricing server responds with the ConsumptionTariffIntervalList. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<ConsumptionTariffIntervalList all="1" href="/tp/3/rc/3/tti/5/cti" results="1" xmlns="urn:ieee:std:2030.5:ns">` `  <ConsumptionTariffInterval href="/tp/3/rc/3/tti/5/cti/1">` `  <consumptionBlock>1</consumptionBlock>` `  <price>113000</price>` `  <startValue>0</startValue>` `  </ConsumptionTariffInterval>` `</ConsumptionTariffIntervalList>`

C.16 Billing: Billing period

The flow diagram in Figure C.15 describes a Billing client device obtaining Billing information including billing period.

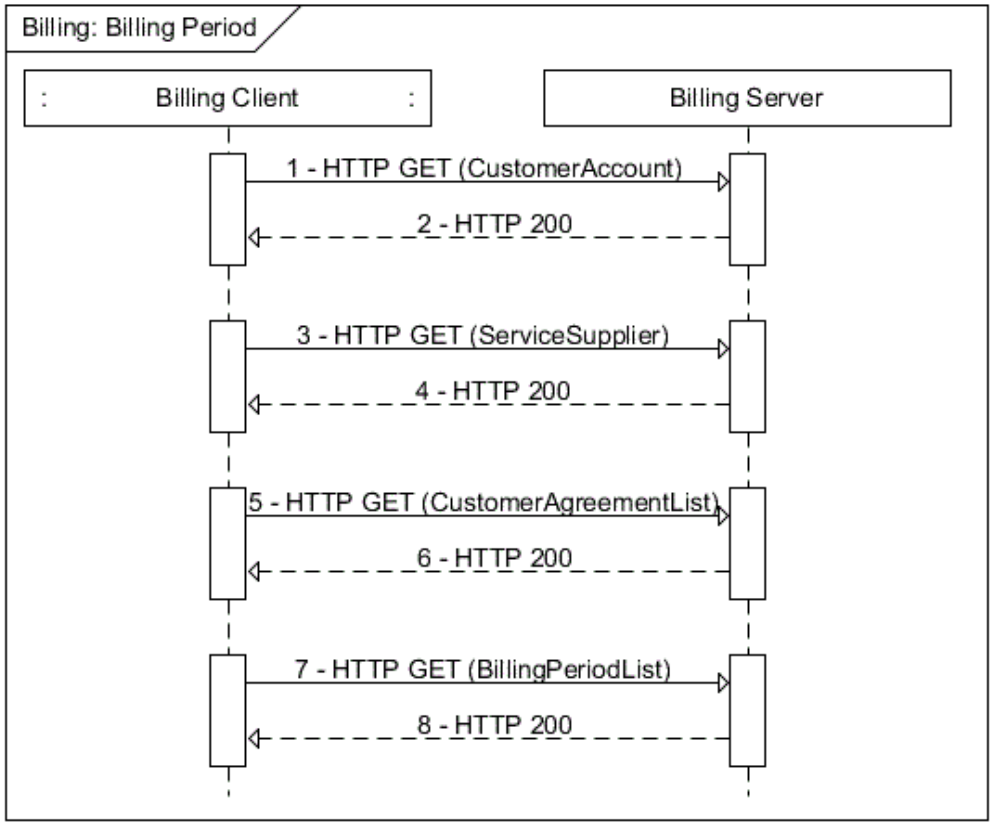


Figure C.15—Billing period

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*.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.17—POX example: Billing period

Step	Description
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`
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}` `<CustomerAccount href="/bill/1" xmlns="urn:ieee:std:2030.5:ns">` `  <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}` `<ServiceSupplier href="/ss" xmlns="urn:ieee:std:2030.5:ns">` `  <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`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.17—POX example: Billing period *(continued)*

Step	Description
6	Billing server responds with the CustomerAgreementList. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<CustomerAgreementList all="1" href="/bill/1/ca" results="1" subscribable="0"` `xmlns="urn:ieee:std:2030.5:ns">` `  <CustomerAgreement href="/bill/1/ca/1">` `  <mRID>65f839fc951345e70000e566</mRID>` `  <description>Electric Service - 4/1/2012</description>` `  <ActiveBillingPeriodListLink all="1" href="/bill/1/ca/1/actbp"/>` `  <BillingPeriodListLink all="1" href="/bill/1/ca/1/bp"/>` `  <HistoricalReadingListLink all="1" href="/bill/1/ca/1/ver"/>` `  <ProjectionReadingListLink all="1" href="/bill/1/ca/1/pro"/>` `  <serviceLocation>Acct. XXX-XXXXX-384 (Elm St.)</serviceLocation>` `  <TariffProfileLink href="/tp/3"/>` `  <UsagePointLink href="/upt/1"/>` `  </CustomerAgreement>` `</CustomerAgreementList>`
7	Client GETs the BillingPeriodList from the Billing server. Client sends the following request: `GET /bill/1/ca/1/bp?l=1 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`
8	Billing server responds with the BillingPeriodList. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<BillingPeriodList all="2" href="/bill/1/ca/1/bp" results="1" subscribable="1"` `xmlns="urn:ieee:std:2030.5:ns">` `  <BillingPeriod href="/bill/1/ca/1/bp">` `  <billLastPeriod>140730000</billLastPeriod>` `  <billToDate>83550000</billToDate>` `  <interval>` `  <duration>2419200</duration>` `  <start>1360195200</start>` `  </interval>` `  <statusTimeStamp>1361577600</statusTimeStamp>` `  </BillingPeriod>` `</BillingPeriodList>`

C.17 Billing: Historical

The flow diagram shown in Figure C.16 describes a Billing client device obtaining historical Billing readings.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

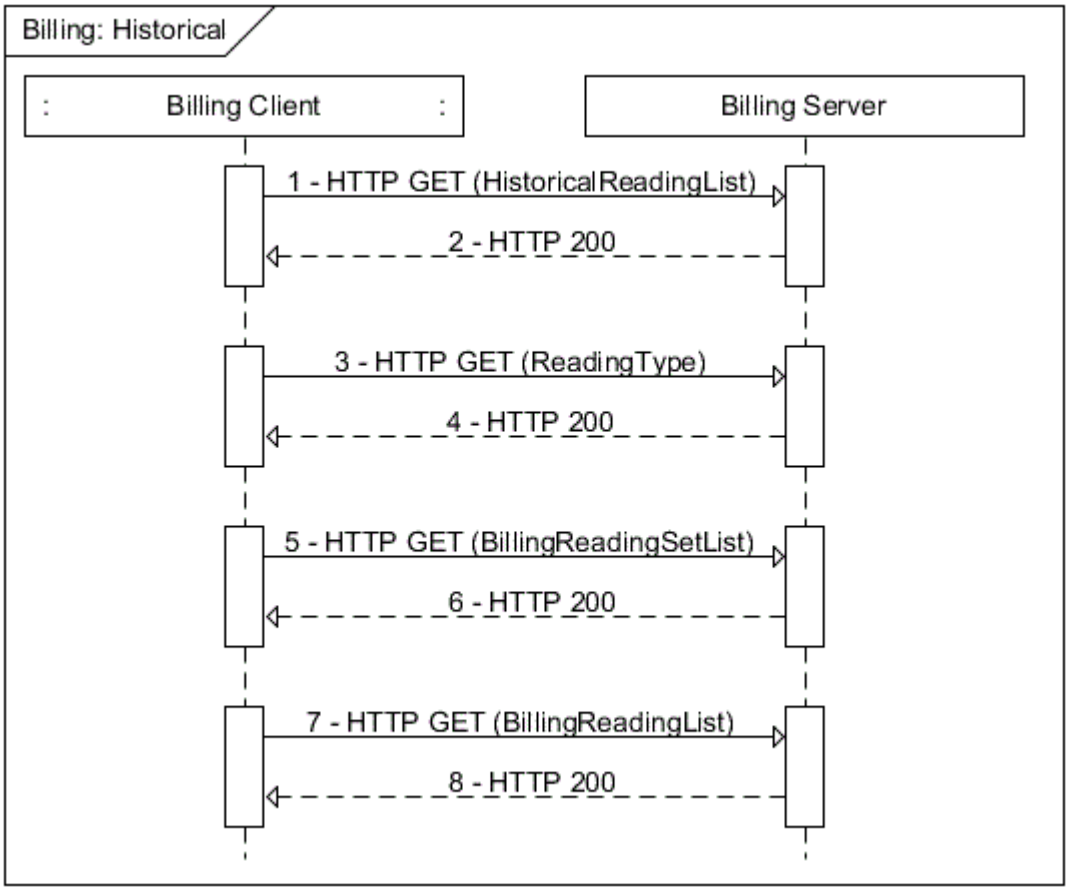


Figure C.16—Billing historical

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.18—POX example: Billing historical

Step	Description
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"` `xmlns="urn:ieee:std:2030.5:ns">` `  <HistoricalReading href="/bill/1/ca/1/ver/1">` `  <mRID>ce7f99f087b5e3de0000e566</mRID>` `  <description>Hourly usage (kWh)</description>` `  <BillingReadingSetListLink all="180" href="/bill/1/ca/1/ver/1/brs"/>` `  <ReadingTypeLink href="/rt/3"/>` `  </HistoricalReading>` `</HistoricalReadingList>`
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`
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="urn:ieee:std:2030.5:ns">` `  <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>`
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`

Copyright © 2018 IEEE. All rights reserved.

Table C.18—POX example: Billing historical *(continued)*

Step	Description
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="urn:ieee:std:2030.5:ns">` `  <BillingReadingSet href="/bill/1/ca/1/ver/1/brs/1">` `  <mRID>82866ecc81c9638a0000e566</mRID>` `  <timePeriod>` `  <duration>86400</duration>` `  <start>1361491200</start>` `  </timePeriod>` `  <BillingReadingListLink all="24" href="/bill/1/ca/1/ver/1/brs/1/br"/>` `  </BillingReadingSet>` `</BillingReadingSetList>`
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}` `<BillingReadingList all="24" href="/bill/1/ca/1/ver/1/brs/1/br" results="1" xmlns="urn:ieee:std:2030.5:ns">` `  <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>`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.18 Billing: Projection

The flow diagram shown in Figure C.17 describes a Billing client device obtaining a Billing projection.

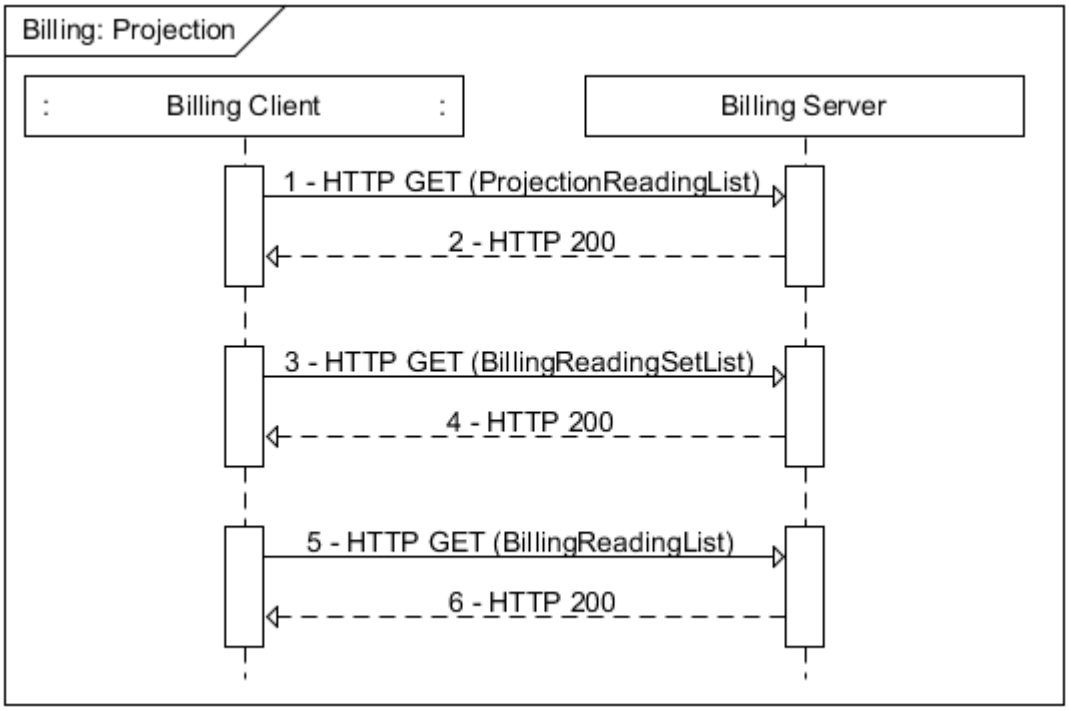


Figure C.17—Billing projection

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.19—POX example: Billing projection

Step	Description
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}` `<ProjectionReadingList all="2" href="/bill/1/ca/1/pro" results="1"` `xmlns="urn:ieee:std:2030.5:ns">` `<ProjectionReading href="/bill/1/ca/1/pro/1">` `<mRID>d0b05a2e65144fca0000e566</mRID>` `<description>Billing Projections</description>` `<BillingReadingSetListLink all="1" href="/bill/1/ca/1/pro/1/brs"/>` `</ProjectionReading>` `</ProjectionReadingList>`
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}` `<BillingReadingSetList all="1" href="/bill/1/ca/1/pro/1/brs" results="1"` `subscribable="1" xmlns="urn:ieee:std:2030.5:ns">` `<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`

Copyright © 2018 IEEE. All rights reserved.

Table C.19—POX example: Billing projection *(continued)*

Step	Description
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}` `<BillingReadingList all="2" href="/bill/1/ca/1/pro/1/brs/1/br" results="1" xmlns="urn:ieee:std:2030.5:ns">` `  <BillingReading href="/bill/1/ca/1/pro/1/brs/1/br/1">` `  <consumptionBlock>2</consumptionBlock>` `  <qualityFlags>40</qualityFlags>` `  <timePeriod>` `  <duration>0</duration>` `  <start>1361664000</start>` `  </timePeriod>` `  <touTier>0</touTier>` `  <value>400000</value>` `  </BillingReading>` `</BillingReadingList>`

C.19 File loading

The flow diagram shown in Figure C.18 describes how a loading device (LD) queries a file server (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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

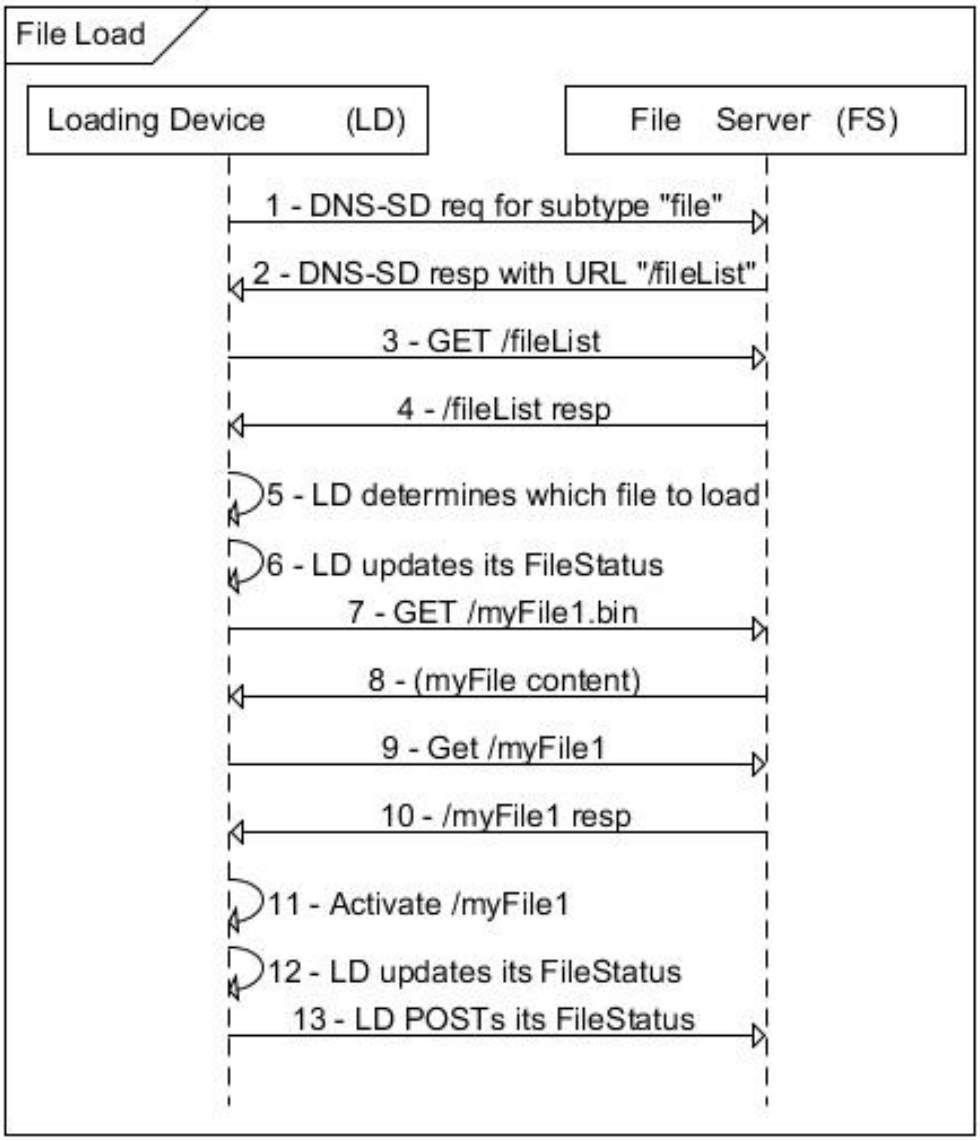


Figure C.18—File load—flow diagram

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.20—File load: Flow description

Step	Description
1	The LD issues discovery requests to locate available FS (DNS-SD subtype “file”).
2	The LD has discovered the FS and obtained the URL of its FileList.
3	The LD queries the FileList to determine if there are any available files to be downloaded to the LD. An example FileList query: `GET /fileList?s=0&l=5&type=0x000&mfId=37244&mfModel=123abc&mfVer=23.47.102 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml` 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: {contentLength}` `<FileList all="2" href="http://host1/fileList" results="2"` `xmlns="urn:ieee:std:2030.5:ns">` `  <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 it has two files which match the initial query for firmware/manufacturer PEN 37244, model “123abc,” and are newer than version 23.47.102. The first entry in the list (mfVer 23.48.1) is the latest version. Note: No activation time was provided.
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).
9	Recall that no activation time was included in the original /myFile1 obtained by the LD from the FileList. Whenever an 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 /myFile1 HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Table C.20—File load: Flow description *(continued)*

Step	Description
10	The FS responds with /myFile1. `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<File href="http://host1/myFile1" xmlns="urn:ieee:std:2030.5:ns">` `  <activateTime>3763784682</activateTime>` `  <fileURI>http://host1/myfile1.bin</fileURI>` `  <mfID>37244</mfID>` `  <mfModel>123abc</mfModel>` `  <mfVer>23.48.1</mfVer>` `  <size>128000</size>` `  <type>00</type>` `</File>` 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.
11	The LD waits until the activation time specified for /myFile1 is reached, then places the file into the activated state. In the case of a firmware file, the file is now the running image.
12	The LD again updates its FileStatus resource.
13	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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.20 Flow Reservation: General

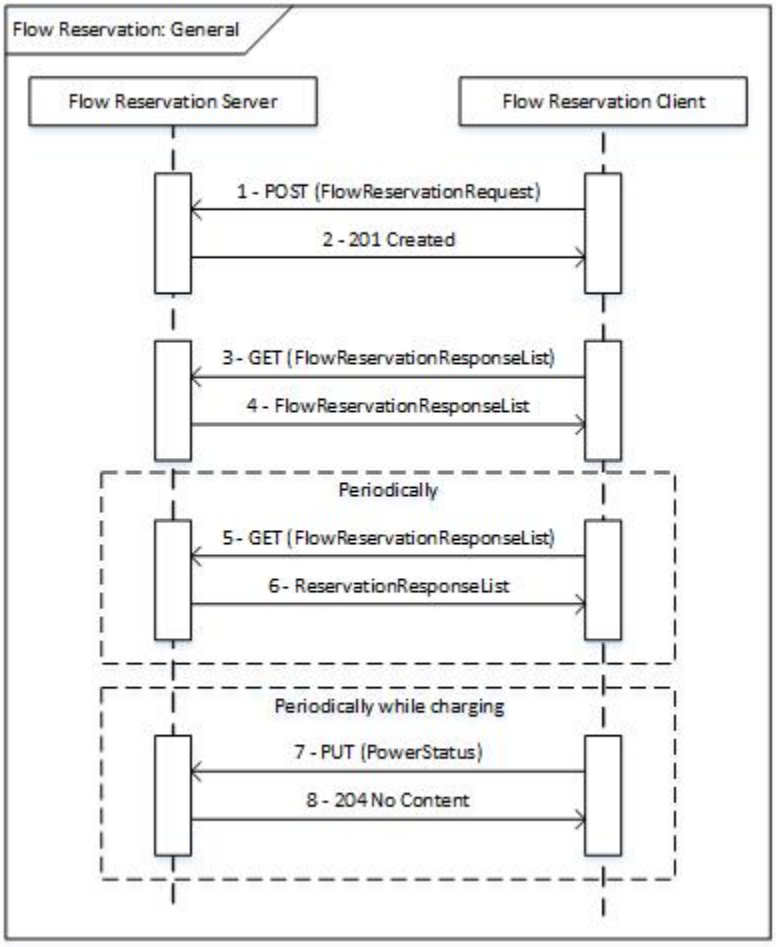


Figure C.19—Flow Reservation: General

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, 12 kWh energy requested at a power level of 7 kW, 7371 seconds duration requested.

— Subsequently, the server creates a FlowReservationResponse with a charge interval between 1:00 AM and 5:20 AM at 3 kW.

Step 3—Client requests the FlowReservationResponseList to find the response matching the request.

Step 4—Server responds with the FlowReservationResponseList.

Step 5—Client periodically requests the FlowReservationResponse to look for changes.

Step 7—Client updates PowerStatus periodically during charging.

Note: In most cases, registration is required to obtain access to request Flow Reservations.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.21—POX example: Flow Reservation—general

Step	Description
1	Client POSTs a FlowReservationRequest to the Flow Reservation server at 22 September 2013, 5:00 PM. Client sends the following request: `POST /edev/3/frq HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FlowReservationRequest xmlns="urn:ieee:std:2030.5:ns">` `<mRID>68512866203db3b10000e566</mRID>` `<description>Charge from 12:00 AM to 8:00 AM</description>` `<creationTime>1379869200</creationTime> <!-- 9/22/2013 5:00 PM -->` `<durationRequested>7371</durationRequested> <!-- 6171sec charging + 1200sec conditioning -->` `<energyRequested>` `<multiplier>3</multiplier> <!-- 12 kWh -->` `<value>12</value>` `</energyRequested>` `<intervalRequested>` `<duration>28800</duration> <!-- 8 hours -->` `<start>1379894400</start> <!-- 9/23/2013 12:00 AM -->` `</intervalRequested>` `<powerRequested>` `<multiplier>3</multiplier> <!-- 7 kW -->` `<value>7</value>` `</powerRequested>` `<RequestStatus>` `<dateTime>1379869200</dateTime>` `<requestStatus>0</requestStatus> <!-- Requested -->` `</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: `GET /edev/3/frp HTTP/1.1` `Host: {hostname}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.21—POX example: Flow Reservation—general *(continued)*

Step	Description
4	Flow Reservation server responds with the FlowReservationResponseList. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FlowReservationResponseList all="1" href="/edev/3/frp" results="1" subscribable="1" xmlns="urn:ieee:std:2030.5:ns">` `  <FlowReservationResponse href="/edev/3/frp/1" subscribable="1">` `  <mRID>f8afa6fde40db98d0000ea75</mRID>` `  <description>Charge from 1:00 AM to 5:20 AM</description>` `  <!-- 1379869260 = 09/22/2013 5:01:00 PM -->` `  <creationTime>1379869260</creationTime>` `  <EventStatus>` `  <currentStatus>0</currentStatus>  <!-- Scheduled -->` `  <dateTime>1379869260</dateTime>` `  <potentiallySuperseded>false</potentiallySuperseded>` `  </EventStatus>` `  <interval>` `  <duration>15600</duration>  <!-- 4 hours 20 minutes -->` `  <start>1379898000</start>  <!-- 1:00 AM 9/23/2013 -->` `  </interval>` `  <energyAvailable>` `  <!-- 12000 Wh -->` `  <multiplier>0</multiplier>` `  <value>12000</value>` `  </energyAvailable>` `  <powerAvailable>` `  <multiplier>0</multiplier>` `  <value>3000</value>` `  </powerAvailable>` `  <subject>68512866203db3b10000e566</subject>` `  </FlowReservationResponse>` `</FlowReservationResponseList>`
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}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.21—POX example: Flow Reservation—general *(continued)*

Step	Description
6	Flow Reservation server responds with the FlowReservationResponseList. Server sends the following response: `HTTP/1.1 200 OK` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FlowReservationResponseList all="1" href="/edev/3/frp" results="1" subscribable="1" xmlns="urn:ieee:std:2030.5:ns">` `  <FlowReservationResponse href="/edev/3/frp/1" subscribable="1">` `  <mRID>f8afa6fde40db98d0000ea75</mRID>` `  <description>Charge from 1:00 AM to 5:20 AM</description>` `  <!-- 1379869260 = 09/22/2013 5:01:00 PM -->` `  <creationTime>1379869260</creationTime>` `  <EventStatus>` `  <currentStatus>0</currentStatus>  <!-- Scheduled -->` `  <dateTime>1379869260</dateTime>` `  <potentiallySuperseded>false</potentiallySuperseded>` `  </EventStatus>` `  <interval>` `  <duration>15600</duration>  <!-- 4 hours 20 minutes -->` `  <start>1379898000</start>  <!-- 1:00 AM 9/23/2013 -->` `  </interval>` `  <energyAvailable>` `  <!-- 12000 Wh -->` `  <multiplier>0</multiplier>` `  <value>12000</value>` `  </energyAvailable>` `  <powerAvailable>` `  <multiplier>0</multiplier>` `  <value>3000</value>` `  </powerAvailable>` `  <subject>68512866203db3b10000e566</subject>` `  </FlowReservationResponse>` `</FlowReservationResponseList>`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.21—POX example: Flow Reservation—general *(continued)*

Step	Description
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}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<PowerStatus xmlns="urn:ieee:std:2030.5:ns">` `<!-- less than LowChargeThreshold remaining -->` `<batteryStatus>2</batteryStatus>` `<!-- 3:00 AM 9/23/2013 -->` `<changedTime>1379905200</changedTime>` `<!-- mains -->` `<currentPowerSource>1</currentPowerSource>` `<estimatedChargeRemaining>150</estimatedChargeRemaining>` `<PEVInfo>` `<chargingPowerNow>` `<multiplier>0</multiplier>` `<value>3000</value>` `</chargingPowerNow>` `<energyRequestNow>` `<multiplier>0</multiplier>` `<value>6100</value>` `</energyRequestNow>` `<maxForwardPower>` `<multiplier>3</multiplier>` `<!-- 24 kW -->` `<value>24</value>` `</maxForwardPower>` `<!-- 3600sec * 6100Wh/7000W + 1200sec conditioning -->` `<minimumChargingDuration>4337</minimumChargingDuration>` `<targetStateOfCharge>10000</targetStateOfCharge>` `<!-- 8:00 AM 9/23/2013 -->` `<timeChargeIsNeeded>1379923200</timeChargeIsNeeded>` `<timeChargingStatusPEV>1379905200</timeChargingStatusPEV>` `</PEVInfo>` `</PowerStatus>`
8	Flow Reservation server responds: `HTTP/1.1 204 No Content`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.21 Flow Reservation: Cancel

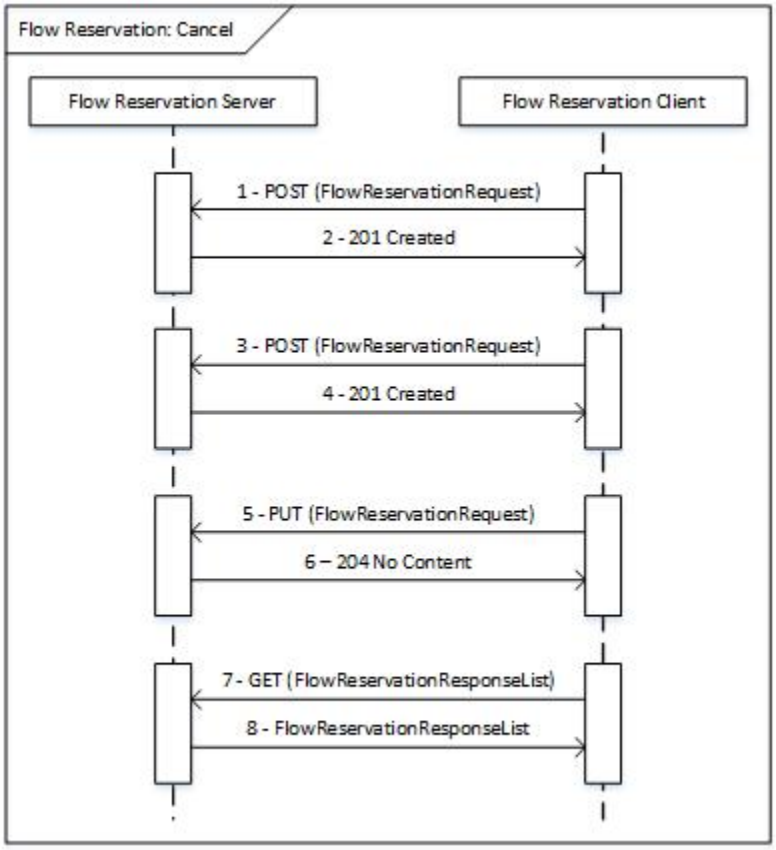


Figure C.20—Flow Reservation: Cancel

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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.22—POX example: Flow Reservation—cancel

Step	Description
1	Client POSTs a FlowReservationRequest to the Flow Reservation server at 22 September 2013, 5:00 PM. Client sends the following request: `POST /edev/3/frq HTTP/1.1` `Host: {hostname}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FlowReservationRequest xmlns="urn:ieee:std:2030.5:ns">` `<mRID>68512866203db3b10000e566</mRID>` `<description>Charge from 12:00 AM to 8:00 AM</description>` `<creationTime>1379869200</creationTime> <!-- 9/22/2013 5:00 PM -->` `<durationRequested>7371</durationRequested> <!-- 6171sec charging + 1200sec conditioning -->` `<energyRequested>` `<multiplier>3</multiplier> <!-- 12 kWh -->` `<value>12</value>` `</energyRequested>` `<intervalRequested>` `<duration>28800</duration> <!-- 8 hours -->` `<start>1379894400</start> <!-- 9/23/2013 12:00 AM -->` `</intervalRequested>` `<powerRequested>` `<multiplier>3</multiplier> <!-- 7 kW -->` `<value>7</value>` `</powerRequested>` `<RequestStatus>` `<dateTime>1379869200</dateTime>` `<requestStatus>0</requestStatus> <!-- Requested -->` `</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 23 September 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}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FlowReservationRequest xmlns="urn:ieee:std:2030.5:ns">` `<mRID>68512866203db3b20000e566</mRID>` `<description>Charge from 1:00 AM to 6:00 AM</description>` `<creationTime>1379898000</creationTime> <!-- 9/23/2013 1:00 AM -->` `<durationRequested>7371</durationRequested> <!-- 6171sec charging + 1200sec conditioning -->` `<energyRequested>` `<multiplier>3</multiplier> <!-- 12 kWh -->` `<value>12</value>` `</energyRequested>` `<intervalRequested>` `<duration>14400</duration> <!-- 5 hours -->` `<start>1379898000</start> <!-- 9/23/2013 1:00 AM -->` `</intervalRequested>` `<powerRequested>` `<multiplier>3</multiplier> <!-- 7 kW -->` `<value>7</value>` `</powerRequested>` `<RequestStatus>` `<dateTime>1379898000</dateTime>` `<requestStatus>0</requestStatus> <!-- Requested -->` `</RequestStatus>` `</FlowReservationRequest>`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.22—POX example: Flow Reservation—cancel *(continued)*

Step	Description
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}` `Content-Type: application/sep+xml` `Content-Length: {contentLength}` `<FlowReservationRequest xmlns="urn:ieee:std:2030.5:ns">` `  <mRID>68512866203db3b10000e566</mRID>` `  <description>Charge from 12:00 AM to 8:00 AM</description>` `  <creationTime>1379869200</creationTime>  <!-- 9/22/2013 5:00 PM -->` `  <durationRequested>7371</durationRequested> <!-- 6171sec charging +` `  1200sec conditioning -->` `  <energyRequested>` `  <multiplier>3</multiplier>  <!-- 12 kWh -->` `  <value>12</value>` `  </energyRequested>` `  <intervalRequested>` `  <duration>28800</duration>  <!-- 8 hours -->` `  <start>1379894400</start>  <!-- 9/23/2013 12:00 AM -->` `  </intervalRequested>` `  <powerRequested>` `  <multiplier>3</multiplier>  <!-- 7 kW -->` `  <value>7</value>` `  </powerRequested>` `  <RequestStatus>` `  <dateTime>1379898060</dateTime>` `  <requestStatus>1</requestStatus>  <!-- Canceled -->` `  </RequestStatus>` `</FlowReservationRequest>`
6	Flow Reservation server sends the following response: `HTTP/1.1 204 No Content`
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}` `Accept: application/sep+xml`

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table C.22—POX example: Flow Reservation—cancel *(continued)*

Step	Description
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 Content-Length: {contentLength} <FlowReservationResponseList all="2" href="/edev/3/frp" results="2" subscribable="1" xmlns="urn:ieee:std:2030.5:ns"> <FlowReservationResponse href="/edev/3/frp/1" subscribable="1"> <mRID>f8afa6fde40db98d0000ea75</mRID> <description>Charge from 1:00 AM to 5:20 AM</description> <creationTime>1379869260</creationTime> <EventStatus> <!-- Canceled --> <currentStatus>2</currentStatus> <dateTime>1379898060</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <!-- 4 hours 20 minutes --> <duration>15600</duration> <!-- 1:00 AM 9/23/2013 --> <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/2" subscribable="1"> <mRID>f8afa6fde40db98e0000ea75</mRID> <description>Charge from 1:02 AM to 4:22 AM</description> <creationTime>1379898120</creationTime> <EventStatus> <!-- Active --> <currentStatus>1</currentStatus> <dateTime>1379898120</dateTime> <potentiallySuperseded>false</potentiallySuperseded> </EventStatus> <interval> <!-- 3 hours 20 minutes --> <duration>12000</duration> <!-- 1:02 AM 9/23/2013 --> <start>1379898120</start> </interval> <energyAvailable> <multiplier>0</multiplier> <value>12000</value> </energyAvailable> <powerAvailable> <multiplier>0</multiplier> <value>4000</value> </powerAvailable> <subject>68512866203db3b20000e566</subject> </FlowReservationResponse> </FlowReservationResponseList>

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.22 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 latter two controlling the randomization behavior. For these examples, we will assume that 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.

C.22.1 Simple event

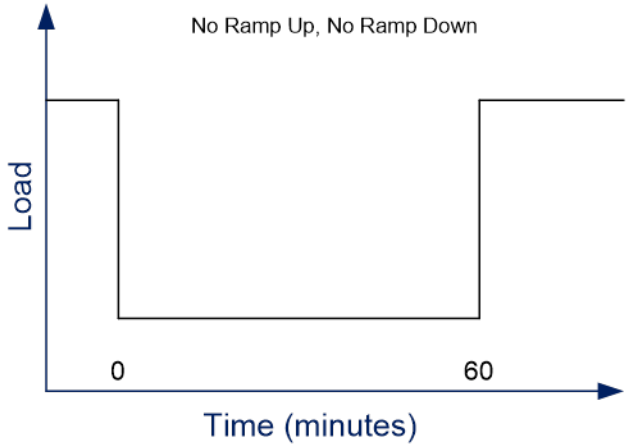


Figure C.21—Simple event, no ramp up, no ramp down

NOTE—This use case is NOT targeted for large population control.

Table C.23—Simple event values

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	0
randomizeDuration	0

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.22.2 Event with positive randomized duration

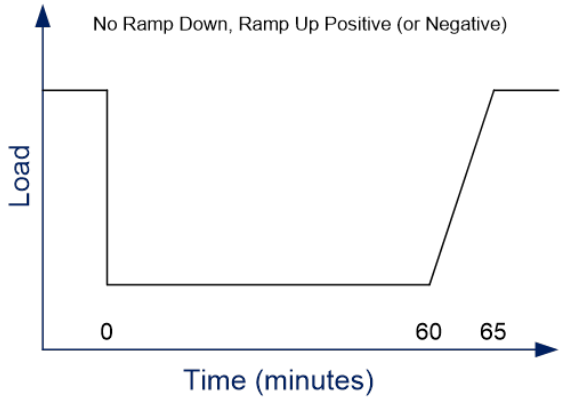


Figure C.22—Event with positive randomization duration

Table C.24—Positive duration randomization values

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	0
randomizeDuration	300

C.22.3 Event with positive randomized start

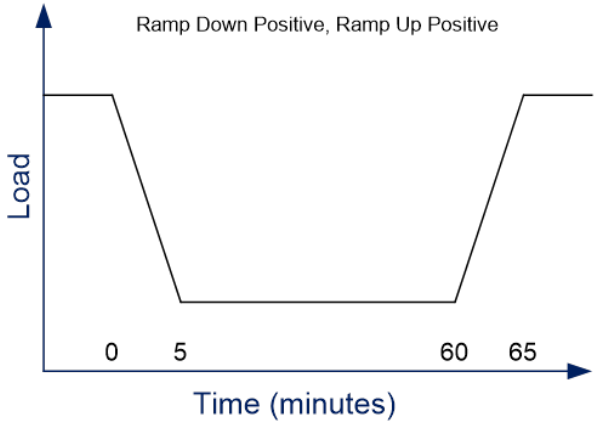


Figure C.23—Event with positive randomization start

Table C.25—Positive start randomization values

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	300
randomizeDuration	0

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.22.4 Event with negative randomized start

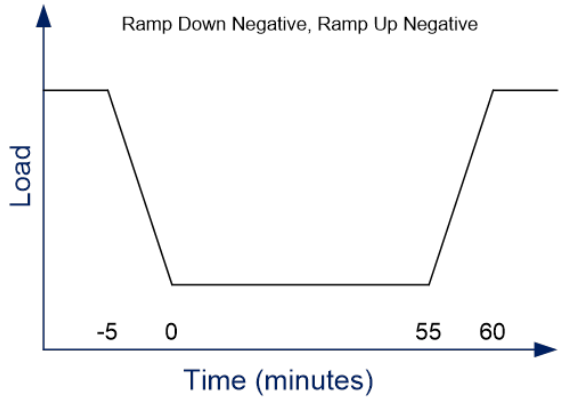


Figure C.24—Event with negative randomized start

Table C.26—Negative start randomization values

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	−300
randomizeDuration	0

C.22.5 Event with positive randomization and finishing in one hour

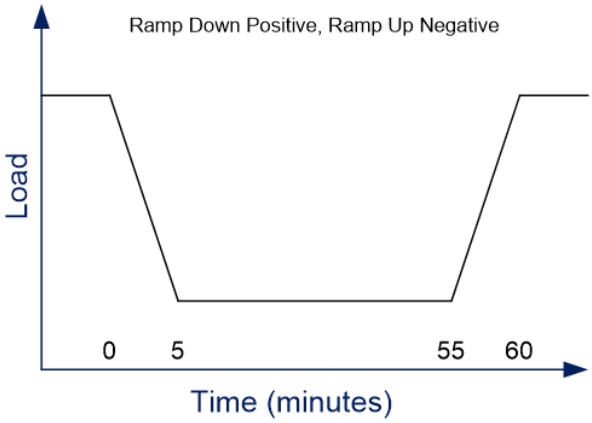


Figure C.25—Positive randomization in an hour

Table C.27—Positive randomization in one hour values

Attribute	Value (seconds)
start	0
duration	3300
randomizeStart	300
randomizeDuration	0

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.22.6 Event with negative randomized start and at least one hour duration

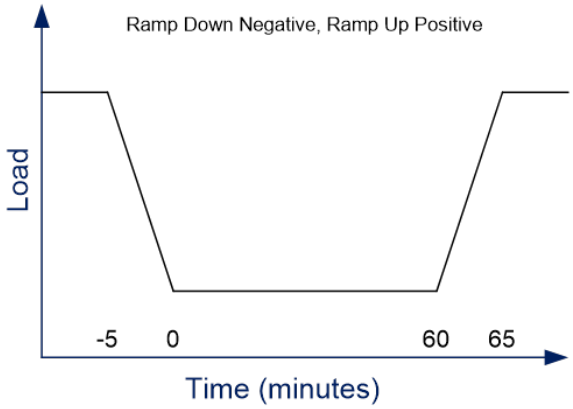


Figure C.26—Event with negative start in one hour

Table C.28—Negative start in one hour values

Attribute	Value (seconds)
start	0
duration	3900
randomizeStart	−300
randomizeDuration	0

C.22.7 Event with randomized start and long ramp up

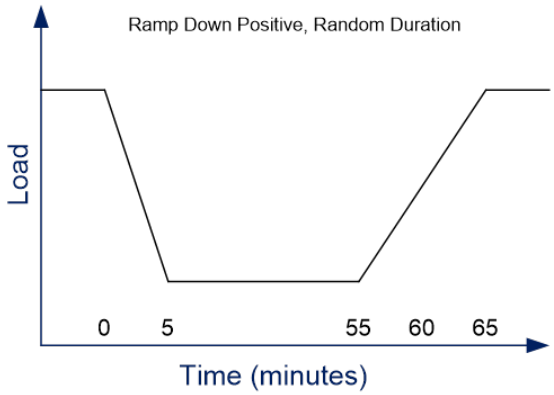


Figure C.27—Randomized start and long ramp up

Table C.29—Randomized start with long ramp up values

Attribute	Value (seconds)
start	0
duration	3600
randomizeStart	300
randomizeDuration	300

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

C.22.8 Event with positive randomized start and fixed end time

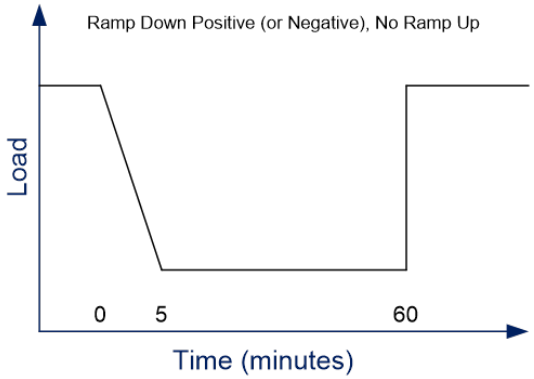


Figure C.28—Randomized start and fixed end time

NOTE—This scenario is NOT possible with the current set of parameters. For grid stability “snap” back on is not a desirable use case.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Annex D

(informative)

Guidelines

D.1 Pricing implementation guidelines

D.1.1 Introduction

This subclause 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 subcluase assumes readers are familiar with the Pricing (10.5) and Metering (10.4) function set text. Readers may also wish to consult the WADL and schema (the supplemental material of IEEE Std 2030.5) for more detailed information.

D.1.2 Implementing common tariff designs

This subclause 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 C.15. 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 (TOU) 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”.

D.1.3 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

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Per the supplemental material of IEEE Std 2030.5, 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 the supplemental material of IEEE Std 2030.5, 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".

D.1.4 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 IEEE Std 2030.5. 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 the table below:

Table D.1—TOU rate

Time period	Description	Price	Tier
Midnight to 8 AM	Off-peak	$0.10	1
8 AM to 10 AM	Mid-peak	$0.20	2
10 AM to 6 PM	On-peak	$0.40	3
6 PM to midnight	Mid-peak	$0.20	2
Intermittent	CPP	$0.70	4

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 IEEE Std 2030.5.

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 IEEE 2030.5 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

— The local time is 9 AM on 16 July 2012.

— There are no CPP events scheduled for the next 48 hours.

In this scenario, the server's TimeTariffIntervalList will contain eight 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".

D.1.5 Consumption-based block rate design

In some jurisdictions (e.g., California, British Columbia, the United Kingdom), 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 D.2.

Table D.2—Consumption-based rates

Block	Consumption startValue	Price
1	0	$0.12
2	150	$0.14
3	250	$0.23
4	300	$0.27
5	350	$0.30

Block numbers are ordered by *startValue* from lowest to highest; this is mandatory in IEEE Std 2030.5. While the associated prices in this example also increase with each block number, this is not mandated by IEEE Std 2030.5. 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 (five, 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.

D.1.6 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.

Copyright © 2018 IEEE. All rights reserved.

For this TOU with consumption-based rate design, assume a tariff structure that combines the previous two examples as shown in Table D.3.

Table D.3—Tariff structure

Time	Tier	Block	Consumption startValue	Description	Price
Midnight to 8 AM	1	1	0	Off-peak	$0.22
		2	150		$0.24
		3	250		$0.33
		4	300		$0.37
		5	350		$0.40
8 AM to 10 AM	2	1	0	Mid-peak	$0.32
		2	150		$0.34
		3	250		$0.43
		4	300		$0.47
		5	350		$0.50
10 AM to 6 PM	3	1	0	On-peak	$0.52
		2	150		$0.54
		3	250		$0.73
		4	300		$0.77
		5	350		$0.80
6 PM to midnight	2	1	0	Mid-peak	$0.32
		2	150		$0.34
		3	250		$0.43
		4	300		$0.47
		5	350		$0.50
Intermittent	4	1	0	CPP	$0.82
		2	150		$0.84
		3	250		$0.93
		4	300		$0.97
		5	350		$1.00

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 16 July 2012.
- There is a CPP event scheduled for 1 PM to 3 PM on 16 July 2012.

In this scenario, the server's TimeTariffIntervalList will contain ten TimeTariffIntervals:

a) One expired event (Tier 1, present day 00:00:00 to 08:00:00)

b) One active event (Tier 2, present day 08:00:00 to 10:00:00)

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

c) Eight scheduled events, corresponding to:

1) Tier 3, present day 10:00:00 to 13:00:00
2) Tier 4 (CPP), present day 13:00:00 to 15:00:00
3) Tier 3, present day 15:00:00 to 18:00:00
4) Tier 2, present day 18:00:00 to next day 00:00:00
5) Tier 1, next day 00:00:00 to 08:00:00
6) Tier 2, next day 08:00:00 to 10:00:00
7) Tier 3, next day 10:00:00 to 18:00:00
8) Tier 2, next day 18:00:00 to day after next 00:00:00

Each TimeTariffInterval links to its own ConsumptionTariffIntervalList, and each ConsumptionTariffIntervalList will contain five 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.

D.1.7 Coordinated and uncoordinated Pricing and Metering servers

As with all function set servers in IEEE Std 2030.5, 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 when customers do not have smart meters, but install their own and link their HAN to their electricity service provider's Pricing server on its website.

This subclause 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). Rule #1 below assumes that the Pricing and Metering servers are coordinated. Rules #2 and #3 are meant to guide implementations in scenarios where the Pricing and Metering servers are uncoordinated.

— Rule #1: 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.

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

— Rule #2: If Rule #1 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

— Rule #3: If Rule #2 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.

D.2 PEV implementation guidelines (subject to work with SAE and ISO/IEC)

Plug-in electric vehicles (PEVs) should implement Registration, Demand Response Load Control (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.

Copyright © 2018 IEEE. All rights reserved.

Annex E

(informative)

Mapping to IEEE Std 1547-2018

This annex maps various parameters in IEEE Std 2030.5-2018 to those needed by IEEE Std 1547-2018. The contents of this annex are not considered to be normative and are provided for reference.

Table E.1—Nameplate information

Parameter	Description	IEEE 2030.5-2018 parameter
Active power rating at unity power factor (nameplate active power rating)	Active power rating in watts at unity power factor	DERCapability::rtgMaxW
Active power rating at specified over-excited power factor	Active power rating in watts at specified over-excited power factor	DERCapability::rtgOverExcitedW
Specified over-excited power factor	Over-excited power factor	DERCapability::rtgOverExcitedPF
Active power rating at specified under-excited power factor	Active power rating in watts at specified under-excited power factor	DERCapability::rtgUnderExcitedW
Specified under-excited power factor	Under-excited power factor	DERCapability::rtgUnderExcitedPF
Apparent power maximum rating	Maximum apparent power rating in voltamperes	DERCapability::rtgMaxVA
Normal operating performance category	Indication of reactive power and voltage/power control capability	DERCapability::rtgNormalCategory
Abnormal operating performance category	Indication of voltage and frequency ride-through capability category I, II, or III	DERCapability::rtgAbnormalCategory
Reactive power injected maximum rating	Maximum injected reactive power rating in vars	DERCapability::rtgMaxVar
Ractive power absorbed maximum rating	Maximum absorbed reactive power rating in vars	DERCapability::rtgMaxVarNeg
Active power charge maximum rating	Maximum active power charge rating in watts	DERCapability::rtgMaxChargeRateW
Apprarent power charge maximum rating	Maximum apparent power charge rating in voltamperes; may differ from the apparent power maximum rating	DERCapability::rtgMaxChargeRateVA
AC voltage nominal rating	Nominal ac voltage rating in rms volts	DERCapability::rtgVNom
AC voltage maximum rating	Maximum ac voltage rating in rms volts	DERCapability::rtgMaxV
AC voltage minimum rating	Minimum ac voltage rating in rms volts	DERCapability::rtgMinV
Supported control mode functions	Indication of support for each control mode function	DERCapability::modesSupported
Reactive susceptance that remains connected to the Area EPS in the cease to energize and trip state	Reactive susceptance that remains connected to the Area EPS in the cease to energize and trip state	DERCapability::rtgReactiveSusceptance
Manufacturer	Manufacturer	DeviceInformation::mfID
Model	Model	DeviceInformation::mfModel
Serial number	Serial number	DeviceInformation::mfSerNum
Version	Version	DeviceInformation::mfHwVer DeviceInformation::swVer

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table E.2—Monitoring information

Parameter	Description	IEEE 2030.5-2018 parameter
Active power	Active power in watts	ReadingType::accumulationBehaviour = 12 ReadingType::commodity = 1 ReadingType::flowDirection = 19 ReadingType::kind = 37 ReadingType::uom = 38
Reactive power	Reactive power in vars	ReadingType::accumulationBehaviour = 12 ReadingType::commodity = 1 ReadingType::flowDirection = 19 ReadingType::kind = 37 ReadingType::uom = 63
Voltage	Voltage(s) in volts (one parameter for single phase systems and three parameters for three phase systems)	ReadingType::accumulationBehaviour = 12 ReadingType::commodity = 1 ReadingType::flowDirection = 1 ReadingType::phase = {phase} ReadingType::uom = 29
Frequency	Frequency in hertz	ReadingType::accumulationBehaviour = 12 ReadingType::commodity = 1 ReadingType::flowDirection = 1 ReadingType::uom = 33
Operational state	Operational state of the DER. The operational state should represent the current state of the DER. The minimum supported states are on and off but additional states may also be supported.	DERStatus::operationalModeStatus
Connection status	Power-connected status of the DER	DERStatus::genConnectStatus DERStatus::storConnectStatus
Alarm status	Active alarm status	DERStatus::alarmStatus
Operational state of charge	0% to 100% of operational energy storage capacity	DERStatus::stateOfChargeStatus

Table E.3—Constant Power Factor mode parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Constant Power Factor mode enable	Enable Constant Power Factor mode	*Active Event*
Constant power factor	Constant power factor setting	DERControl::opModFixedPFInjectW
Constant power factor excitation	Constant power factor excitation setting	DERControl::opModFixedPFInjectW.excitation

Copyright © 2018 IEEE. All rights reserved.

Table E.4—Voltage-reactive power mode parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Voltage-reactive power mode enable	Enable voltage-reactive power mode	*Active Event*
V_{Ref}	Reference voltage	DERCurve::vRef
Autonomous V_{Ref} adjustment enable	Enable/disable autonomous V_{Ref} adjustment	DERCurve::autonomousVRefEnable
V_{Ref} adjustment time constant	Adjustment range for V_{Ref} time constant	DERCurve::autonomousVRefTimeConstant
V/Q curve points	Voltage-reactive power curve points	DERControl::opModVoltVar::CurveData
Open loop response time	Time to ramp up to 90% of the new reactive power target in response to the change in voltage	DERControl::opModVoltVar::openLoopTms

Table E.5—Active power-reactive power mode parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Active power-reactive power mode enable	Enable active power-reactive power mode	*Active Event*
P/Q curve points	Active power-reactive power curve points	DERControl::opModWattVar::CurveData

Table E.6—Constant reactive power mode parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Constant reactive power mode enable	Enable constant reactive power mode	*Active Event*
Constant reactive power	Constant reactive power setting	DERControl::opModFixedVar DERControl::opModTargetVar

Table E.7—Voltage-active power mode parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Voltage-Active Power mode enable	Enable Voltage-Active Power mode	*Active Event*
V/P curve points	Voltage-active power curve points	DERControl::opModVoltWatt::CurveData
Open loop response time	Time to ramp up to 90% of the new active power target in response to the change in voltage	DERControl::opModVoltWatt::openLoopTms

Table E.8—Voltage trip parameters

Parameter	Description	IEEE 2030.5-2018 parameter
High voltage (HV) trip curve points	High voltage shall trip curve points	DERControl::opModHVRTMustTrip::CurveData
Low voltage (LV) trip curve points	Low voltage shall trip curve points	DERControl::opModLVRTMustTrip::CurveData

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table E.9—Momentary cessation parameters

Parameter	Description	IEEE 2030.5-2018 parameter
HV momentary cessation curve points	High voltage momentary cessation curve points	DERControl::opModHVRTMomentaryCessation::CurveData
LV momentary cessation curve points	Low voltage momentary cessation curve points	DERControl::opModLVRTMomentaryCessation::CurveData

Table E.10—Frequency parameters

Parameter	Description	IEEE 2030.5-2018 parameter
High frequency (HF) trip curve points	High frequency shall trip curve points	DERControl::opModHFRTMustTrip::CurveData
Low frequency (LF) trip curve points	Low frequency shall trip curve points	DERControl::opModLFRTMustTrip::CurveData

Table E.11—Frequency droop parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Over-frequency droop DB_{OF}	Frequency droop dead band for over-frequency conditions	DERControl::opModFreqDroop::dBOF
Under-frequency droop DB_{UF}	Frequency droop dead band for under-frequency conditions	DERControl::opModFreqDroop::dBUF
Over-frequency droop K_{OF}	Frequency droop per-unit frequency change for over-frequency conditions corresponding to one per-unit power output change	DERControl::opModFreqDroop::kOF
Under-frequency droop K_{UF}	Frequency droop per-unit frequency change for under-frequency conditions corresponding to one per-unit power output change	DERControl::opModFreqDroop::kUF
Open loop response time	The duration from a step change in control signal input until the output changes by 90% of its final change, before any overshoot	DERControl::opModFreqDroop::openLoopTms

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Table E.12—Enter service after trip parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Permit service	Able to enter or stay in service	DERControl::opModEnergize
Enter service (ES) voltage high	Enter service voltage high	DERSettings::setESHighVolt DefaultDERControl::setESHighVolt
ES voltage low	Enter service voltage low	DERSettings::setESLowVolt DefaultDERControl::setESLowVolt
ES frequency high	Enter service frequency high	DERSettings::setESHighFreq DefaultDERControl::setESHighFreq
ES frequency low	Enter service frequency low	DERSettings::setESLowFreq DefaultDERControl::setESLowFreq
ES delay	Enter service delay	DERSettings::setESDelay DefaultDERControl::setESDelay
ES randomized delay	Enter service randomized delay	DERSettings::setESRandomDelay DefaultDERControl::setESRandomDelay
ES ramp rate	Enter service ramp rate	DERSettings::setESRampTms DefaultDERControl::setESRampTms

Table E.13—Limit maximum active power parameters

Parameter	Description	IEEE 2030.5-2018 parameter
Limit active power enable	Enable mode	*Active Event*
Maximum active power	Maximum active power setting	DERControl::opModMaxLimW

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Annex F

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] CableLabs, Data-over-cable service interface specifications (DOCSIS) (https://www.cablelabs.com/specs/).

[B2] EPRI, Common Functions for Smart Inverters. Product ID: 3002008217.[1]

[B3] Fielding, R. T., "Architectural Styles and the Design of Network-based Software Architectures." Ph.D. diss., University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm).

[B4] IEC 61850, Communication networks and systems in substations—All parts.[2]

[B5] IETF RFC 3535, IAB Network Management Workshop (http://tools.ietf.org/html/rfc3535).

[B6] IETF RFC 3635, Definitions of Management Objects for the Internet-Like Interface Types (http://tools.ietf.org/html/rfc3635).

[B7] IETF RFC 4293, Management Information Base for the Internet Protocol (IP) (http://tools.ietf.org/html/rfc4293).

[B8] IETF RFC 6762, Multicast DNS (http://tools.ietf.org/html/rfc6762).

[B9] Linux (https://www.linux.com/).

[B10] SunSpec, SunSpec Alliance Inverter Control Model, work in progress (https://sunspec.org/).

[1] EPRI publications are available from the Electric Power Research Institute (http://www.epri.com).
[2] IEC publications are available from the International Electrotechnical Commission (http://www.iec.ch) and the American National Standards Institute (http://www.ansi.org/).

Copyright © 2018 IEEE. All rights reserved.

Copyrighted material licensed to Grant Traynor on 2019-04-04 for licensee's use only.
Copyrighted and Authorized by IEEE. Restrictions Apply.

Consensus

WE BUILD IT.

Connect with us on:

Facebook: https://www.facebook.com/ieeesa

Twitter: @ieeesa

LinkedIn: http://www.linkedin.com/groups/IEEESA-Official-IEEE-Standards-Association-1791118

IEEE-SA Standards Insight blog: http://standardsinsight.com

YouTube: IEEE-SA Channel

IEEE
standards.ieee.org
Phone: +1 732 981 0060 Fax: +1 732 562 1571
© IEEE