OPENADE 1.0 SERVICE DEFINITION - REST EXTENSION

VERSION: ***DRAFT V0.92***

Release Date: 7/28/2010

Acknowledgements

The following individuals and their companies have contributed and/or provided support to the work of the OpenADE 1.0 Service Definition - REST Extension:

- Charles Spirakis from Google
- Dave Mollerstuen from Tendril Networks
- Gerald Gray from CIMple Integrations
- Jeffrey Kenward from DTE Energy
- Jeremy McDonald from SCE
- Mark Ortiz from Consumers Energy
- Shawn Hu from Xtensible Solutions / SCE
- Steve Van Ausdall from Xtensible Solutions / SCE

The OpenADE Task Force wishes to thank all of the contributors to OpenADE, especially the above-mentioned individuals and their companies for their support of this important endeavor, as it sets a key foundation for an interoperable Smart Grid.

16 # Document History

17 ## Revision History

18 Date of this revision: July, 28, 2010

Revision Number	Revision Date	Revision By	Summary of Changes	Changes marked
0.5	2/25/10	Steve Van Ausdall	Initial draft discussion version.	N
0.6	3/1/10	Steve Van Ausdall	Additional details about defined resources	N
0.8	4/8/10	Steve Van Ausdall	Broke REST parts out of Common doc	N
0.85	4/15/10	Steve Van Ausdall	Changes from first review meeting	N
0.9	4/22/10	Steve Van Ausdall	Updates from commenters	Y
0.92	7/28/10	Wayne Dennison Steve Van Ausdall	Additional Cleanup and Updates from F2F meeting and Review	N

19 ## Open Issues Log

20 Last updated: Mar. 1, 2010

Issue	Issue Date	Provided By	Summary of the Issue

21

Contents

List of Tables

1 INTRODUCTION

This document contains only the extensions necessary to the OpenADE Common specification to build an AtomPub resource representation syndication server. It is based heavily on GData, an open specification of AtomPub extensions required for general-purpose data synchronization.

These extensions define a collection of resource feeds as a discoverable, stateless data service, using HTTPS to send and receive requests and information in AtomPub XML. This resource-oriented architecture is proposed, similar to efforts elsewhere, such as web / internet of things, GData, and OData. This architecture provides secure access to scalable methods and data resources hosted by the provider, while maintaining concurrency and integrity. All data is secured at the user level, so that access to individual operations can be provided or revoked to external services, and other users' data will still be protected.

1.1 RIGHTS / MANAGEMENT / GOVERNANCE

1.1.1 INTELLECTUAL PROPERTY RIGHTS

This document and the information contained herein is provided on an "AS IS" basis. UCAIug DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

UCAIug requests any party that believes it has a patent claim that would necessarily be infringed by implementations of this UCAIug work, to notify UCAIug immediately, so that fair and reasonable licensing terms can be negotiated. UCAIug invites any party aware of applicable undisclosed patent claims to contact the UCAIug. UCAIug may include such claims on its website, but disclaims any obligation to do so.

UCAIug takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this UCAIug recommendation, can be obtained from the UCAIug. UCAIug makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

1.1.2 CIM OBJECT MODELS

The recommendations herein build on work owned by the IEC. Required extensions identified in this recommendation will be submitted to the IEC, and will be tracked for inclusion in the model.

Information on the management of rights and governance can be found at the page below.
http://www.iec.ch/tctools/patent-guidelines.htm

1.1.3 SERVICE RESOURCE DEFINITIONS

If necessary, UCAIug is willing to work with standards development organizations to incorporate additional aspects of this recommendation into standards, including the specification of how to use profiled (restricted) CIM objects within a RESTful HTTP environment, and possibly the resource definitions themselves.

1.2 REFERENCED SPECIFICATIONS

- [1] OpenADE B&UR 1.0 - http://osgug.ucaiug.org/sgsystems/OpenADE/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fsgsystems%2fOpenADE%2fShared%20Documents%2fBusiness%20and%20User%20Requirements
- [2] OpenADE SRS 1.0 - http://osgug.ucaiug.org/sgsystems/OpenADE/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fsgsystems%2fOpenADE%2fShared%20Documents%2fSRS
- [3] IEC CIM (TC 57 61968/61970) - http://tc57.iec.ch
- [4] OAuth - http://tools.ietf.org/html/draft-hammer-oauth-10
- [5] HTTP – IETF RFC 2616 - http://www.ietf.org/rfc/rfc2616.txt
- [8] PubSubHubbub - http://code.google.com/p/pubsubhubbub/
- [9] Atom Publishing Protocol (RFC 5023) – http://tools.ietf.org/html/rfc5023
- [10] OpenSG OpenADE SD – Common http://osgug.ucaiug.org/sgsystems/OpenADE/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fsgsystems%2fOpenADE%2fShared%20Documents%2fService%20Definition%2fOpenADE%201%2e0%20Service%20Definition

1.3 REFERENCED GUIDANCE

- [G1] 3PDA – Security Profile for Third Party Data Access (ASAP-SG) http://osgug.ucaiug.org/utilisec/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2futilisec%2fShared%20Documents%2fThird%20Party%20Data%20Access%20Security%20Profile
- [G2] GData - http://code.google.com/apis/gdata/docs/2.0/reference.html
- [G3] OData - http://www.odata.org/docs/[MC-APDSU].htm

1.4 NAMESPACES

This document does not define any namespaces. Namespaces defined in referenced specifications shall be used as defined.

2 RESOURCES

Resource Oriented Architecture is nothing new; in fact the web we are all familiar with today provides restful (browser) access to internet resources. When you specify the Address URL of a page, you are providing the address of that resource that you requested. This architecture provides similar operations for external data consumer applications to request data and methods. Generally, data is made available as a feed, which is an agreement about how to query, create, update, request, and delete entries (individual object records, which have a defined schema according to their type).

Since this document is the first to define the general-purpose conventions, several resources were identified to allow consumers to gain access to the resources they want. These extensions are listed below.

- **Service** – To verify service status
- **Enrollment** – To register for access and notifications
- **ActivityRecord** – To provide information about an activity (request)

In addition, addressable resources are defined for the following objects, defined in [10] OpenADE SD - Common.

- **Customer**
- **CustomerAgreement**
- **ServiceDeliveryPoint**
- **MeterReading**
- **IntervalReading**
- **ReadingType**

2.1 URI FORMAT / SYNTAX

The URIs of the resources defined in OpenADE take the general form below. The <baseURL> does not need to be the same across different implementations, since resource addresses include the entire string.

```
http://<baseURL>/<resource>?<query>
```

Resource requests require inputs of user, key, and resource object. Additional path elements may be required for some resources. Resources return a list (feed) or an individual entry.

Requests for protected resources require https, and require authorization token in HTTP header.

The query section contains additional inputs that can be specified to affect processing, passed as a list of `name=value` pairs.

2.2 MESSAGE DOCUMENT FORMAT

Message documents shall use the extended version of the Atom Publishing Protocol ([9] AtomPub, which extends the Atom Syndication Protocol) defined in [G2] GData for the elements described in this document, to fulfill this need.

In addition to the recommended format, it is possible to support additional representations. An input can be accepted to provide RSS or other formats, but these additional formats are all optional, and will only be considered for this specification if needed.

2.3 PAYLOAD ENTITIES

Payload entities will be specializations (subclasses) of the message document “entry”. They will therefore inherit all elements defined in the message document entry, as well as implement an XML schema representation of a CIM class defining additional schema elements.

Feed payloads will contain a list of references to resource entries that match the request query criteria.

The batch payload defined in the Common document allows a number of documents or resource requests to be included in a single request.

2.3.1 RESOURCES

Domain data objects build on the IEC CIM model. In general, resources will be named using the CIM class as the resource part of the URI. Collections are returned if no specific entry ID is specified in the resource path. For listings of fields, see the details for each resource, defined in Common [10].

Examples are shown below. Note that these are separate examples of a GET request issued to the resource path in the comment before the example result.

Customer

```
<!-- /c/12345678910 -->

<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://osgug.ucaiug.org/ns/2010/06/oade">
  <id>12345678910</id>
  <m:CustomerAgreement href="https://{IP Address}/c/12345678910/ca"/>
</entry>
```

CustomerAccount

```
<!-- /c/12345678910/ca/56421587 -->
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://osgug.ucaiug.org/ns/2010/06/oade">
  <id>56421587</id>
  <m:CustomerAuthorisation>
    <m:validityInterval>
      <m:end>2011-12-17T00:00:00Z</m:end>
      <m:start>2010-12-17T00:00:00Z</m:start>
    </m:validityInterval>
  </m:CustomerAuthorisation>
  <m:ServiceDeliveryPoint href="https://{IP Address}/c/12345678910/ca/56421587/sdp"/>
</entry>
```

ServiceDeliveryPoint

```
<!-- /c/12345678910/ca/56421587/sdp/85945261 -->
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://osgug.ucaiug.org/ns/2010/06/oade">
  <id>85945261</id>
  <m:MeterAsset>
    <m:ID>19283746</m:ID>
  </m:MeterAsset>
  <m:MeterReading href="https://{IP Address}/c/12345678910/ca/56421587/sdp/85945261/mr"/>
  <m:name>Guest House</m:name>
  <m:ServiceCategory>
    <m:kind>electricity</m:kind>
  </m:ServiceCategory>
</entry>
```

MeterReading

```
<!-- /c/12345678910/ca/56421587/sdp/85945261/mr/1 -->
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://osgug.ucaiug.org/ns/2010/06/oade">
  <id>1</id>
  <m:IntervalReading href="https://{IP Address}/c/12345678910/ca/56421587/sdp/85945261/mr/1/ir"/>
  <m:ReadingType>
    <m:ID>1001</m:ID>
  </m:ReadingType>
</entry>
```

IntervalReading

```
<!-- /c/12345678910/ca/56421587/sdp/85945261/mr/1/ir/2010-12-17T11_00_00Z -->
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://osgug.ucaiug.org/ns/2010/06/oade">
  <m:endTimeStamp>2010-12-17T11:00:00Z</m:endTimeStamp>
  <m:ReadingQuality>
    <m:quality>interpolated</m:quality>
  </m:ReadingQuality>
  <m:timeStamp>2010-12-17T10:00:00Z</m:timeStamp>
  <m:value>0.0035</m:value>
</entry>
```

ReadingType

```
<!-- /rt/1001 -->
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://osgug.ucaiug.org/ns/2010/06/oade">
  <id>1001</id>
  <defaultQuality>validated</defaultQuality>
  <direction>delivered</direction>
  <kindReading>energy</kindReading>
  <multiplier>k</multiplier>
  <name>Energy Delivered kWh</name>
  <unitSymbol>Wh</unitSymbol>
</entry>
```

3 PATTERNS

This section contains guidance and decisions on how message exchanges flow for the general scenarios below. In general, the constructs and operations defined in [9] AtomPub shall be used, including requests for Services, Workspaces, Collections, Members, Categories, and Media Types. Extensions are generally refined subsets of the full specifications detailed in [G2] GData, and full implementations should not break clients who only implement these recommendations.

3.1 CREATING, UPDATING, DELETING

The POST method is to be used for creation of new entries, PUT is to be used for updates to existing entries, and DELETE is to be used to delete an entry.

3.2 QUERY, REQUEST AND RESPONSE (RETRIEVE) FORMATS

This section specifies the input parameters that can be passed to GET method operations for format, category, reference expansion, sorting, filtering, and iteration through list items.

3.2.1 FORMAT

The default, and only required format, will be CIM-extended AtomPub feed / entry XML.

3.2.2 CATEGORY

Specification of the category of entries is not included in this recommendation. However it can be accomplished using the Atom element "term", and if needed could be supported as a qualifier in queries by accepting category terms as inputs. In general, each CIM object class would be a category of entry, so that the representations of entries can be specified with a schema.

3.2.3 REFERENCE EXPANSION

By default, feed queries will return a list of resource links. If the reference expansion flag is set, entries returned will be expanded to contain their full representation.

(Need to determine if nested expansion is necessary / possible, and if so, how to specify to what level)

[G3] OData uses an `m:inline` extension to the `atom:link` element for this purpose.

3.2.4 SORTING

Ability to specify the sort order of resulting query / request entries is not necessary - subsequent processing of received data can display or rearrange data however desired. However, ordering of entries shall remain consistent across requests, so that an iterator can be used to page through results. Collection results shall be sorted by date of last modification in descending order. (Latest to earliest updated)

3.2.5 FILTERING

Filtering requires inputs that allow the specification of the resource name and/or path, as well as a range of publication or update date/times. Properties of the entry element (defined by its category type) could be defined to be acceptable by default as filter terms for the associated resource. Need to determine if it is feasible to implement all, or if identification is necessary of only the filter terms required for specific use cases. Possibly usage patterns could determine the need for indexing, etc.

If a specific entry ID is specified, that entry is returned.

3.2.6 ITERATION

Iteration inputs allow consumers to request a subset of entries or references at a time, and then page through them for processing. Inputs include the starting entry index, and number of entries per page.

Query results shall include the additional openSearch terms as defined in [G2] GData for iteration.

3.2.7 CONDITIONAL RETRIEVAL

If-None-Match in HTTP header shall be used to retrieve entries only if they have changed as defined in both [G3] OData and [G2] GData.

3.3 EVENT NOTIFICATION (PUB/SUB)

The publish / subscribe pattern is incredibly useful, and is specified mostly in [9] AtomPub. However, there is no mechanism defined in AtomPub to notify subscribers of new feed entries. This requires them to "poll" for new data, and while this is sufficient in many cases, some business processes require ability to notify in order to achieve reduced latency in client updates.

[8] PubSubHubbub defines a mechanism for this purpose, and may be implemented for this purpose.

3.4 BATCH TRANSFERS

If desired, a feed for each data service consumer could be provided through which all subscribed content would be returned in a single request (or series of large chunks). This mechanism should allow any resource type to be included within a single feed. The regular synchronization behavior shall be implemented as a regular feed request or subscription notification with reference expansion specified, as defined in [G2] GData.

4 DISCOVERY

Discovery of available resources may utilize the [9] AtomPub constructs defining services, workspaces, and collections. This is accomplished with a client request to GET the definition of all collections, followed by enrollment / authorization, and finally subscription to the appropriate feeds.

5 METADATA

A "Resource" resource may allow retrieval of the representation of all available resources, and the currently implemented version of each.

[9] AtomPub defines a "workspace collection" for this.

6 EXTENSIBILITY

[9] AtomPub is specified to be extensible, and implementations should be able to function even with different versions of client or server. In addition, section 6.2 in AtomPub provides recommended behavior.

Extensions to the CIM objects will be associated with specific versions of the namespace, specified in the version attribute of the schema element.

7 VERSIONING

As additional capabilities are added to the interface definition, a specification of the version of the definition will be needed to help in service discovery negotiation. This should not change the namespace of any definitions.

8 CONCURRENCY

In most cases, Resources being utilized are Read Only. Should a need for an update to a Resource be determined; In order to ensure data integrity, clients may only update resources if they are updating the current version of the resource. If an update request fails due to conflict (not current version), the client must request the latest version, apply changes to that representation, and retry the update.

[G2] GData and [G3] OData both use ETags for versioning / concurrency management, and are largely compatible. Additional details will be specified as identified during initial implementations.

See GData Resource Versioning http://code.google.com/apis/gdata/docs/2.0/reference.html#ResourceVersioning for more information.

9 RESOURCE DEFINITIONS

The following tables provide an overview of the service resources to be implemented by provider and consumer (3rd Party). Full definitions and expected behavior will be developed in a subsequent publication.

9.1 PROVIDER (UTILITY) RESOURCES

Resource	Method	Inputs (Resource, …)	Outputs	Description
Service	GET	ResourceList	ServiceStatus	Synchronously check connectivity and current operational status of the service
Service	POST	ResourceList	RequestStatus	Asynchronously check connectivity and current operational status of the service
Service	POST	ServiceStatus	RequestStatus	Receive result of status check initiated by Utility
Enrollment	POST	Customer, Key, ResourceList	ActivityRecord	Initiate authorization of 3rd Party customer to receive Utility customer resources
Enrollment	POST	Customer, ResourceList	ActivityRecord	Notify Utility of new authorization completion (future)
Enrollment	POST	Customer, ResourceList	ActivityRecord	Initiate cancel authorization of customer resources
Enrollment	POST	Customer, ResourceList	ActivityRecord	Notify Utility of authorization cancellation
ActivityRecord	GET	ID	ActivityRecord	Receive status of an asynchronous request from Utility
Resource	GET	Format	Resource	Transfer customer usage information data (or other resources, future)
ActivityRecord	POST	ResourceList	RequestStatus	Notify Utility of current status of pending transfers

Table 1: Provider Resource Operations

9.2 CONSUMER (3RD PARTY) RESOURCES

Resource	Method	Inputs (Resource, …)	Outputs	Description
Service	GET	ResourceList	ServiceStatus	Synchronously check connectivity and current operational status of the service
Service	POST	ResourceList	RequestStatus	Asynchronously check connectivity and current operational status of the service
Service	POST	ServiceStatus	RequestStatus	Receive result of status check initiated by 3rd Party
Enrollment	POST	Customer, Key, ResourceList	ActivityRecord	Initiate authorization of Utility customer to receive 3rd Party customer resources (future)
Enrollment	POST	Customer, ResourceList	ActivityRecord	Notify 3rd Party of new authorization completion (future)
Enrollment	POST	Customer, ResourceList	ActivityRecord	Initiate cancel authorization of customer resources
Enrollment	POST	Customer, ResourceList	ActivityRecord	Notify 3rd Party of authorization cancellation
ActivityRecord	GET	ID	ActivityRecord	Receive status of an asynchronous request from 3rd Party
Resource	POST	ResourceList	RequestStatus	Notify 3rd Party that resources were created or updated
Resource	POST	ID	RequestStatus	Notify 3rd Party that new and updated resource files are available

Table 2: Consumer Resource Operations

9.3 RESOURCE DEFINITION

All resources support the patterns in Section 3, returning a (possibly expanded) collection of data resource stream entries. Individual entries can be managed using POST, UPDATE, and DELETE. Permissions may be set according to policy, but guidance is provided regarding the typical configuration in resource details following this table.

9.4 RESOURCE DETAILS

Many of the resources below are necessary to support initial setup and authorization. Implementations shall conform to referenced specifications for details on these interfaces. Clarifications and refinements made to support these service resources are denoted where necessary.

All resources are to be implemented as collections, with the elements listed in the sections below. Additional allowed values may be specified, and will be included here as necessary.

9.4.1 COLLECTION (FEED)

Collections consist of feeds, and exhibit behavior as defined in [9] AtomPub.

9.4.2 AUTHORIZATION

The "Enrollment" resources shall use OAuth 2.0 definitions to allow user owners of provider data resources to grant authorization to 3rd Party for access.