OPENADE 1.0 SERVICE DEFINITION - WEB SERVICES EXTENSION

VERSION: ***DRAFT V0.9***

Release Date: 8/6/2010

Acknowledgements

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

- Chad Maglaque from Microsoft
- Dave Mollerstuen from Tendril Networks
- Gerald Gray from CIMple Integrations
- Mark Ortiz from Consumers Energy
- Shawn Hu from Xtensible Solutions / SCE /Consumers Energy
- 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.

17 # Document History

18 ## Revision History

19 Date of this revision: July 28, 2010

Revision Number	Revision Date	Revision By	Summary of Changes	Changes marked
0.1	4/8/10	Gerald R. Gray	Initial draft discussion version.	N
0.2	4/14/10	Gerald R. Gray	Added example wsdls and xsds provided by Shawn Hu; example SOAP envelope structure	N
0.3	4/15/10	Steve Van Ausdall	Additional cleanup and updates	N
0.4	4/20/10	Steve Van Ausdall	Changes from reviews with SD team	N
0.5	4/20/10	Gerald R. Gray	Added reference to previous AMI-ENT work; additional clean-up from team discussion	N
0.6	4/22/10	Shawn Hu & Mark Ortiz	Added detailed WSDL information	N
0.8	7/28/10	Wayne Dennison Steve Van Ausdall	Additional Cleanup and Updates from F2F meeting and Review	N
0.9	8/6/10	Steve Van Ausdall	Addition of authorization issue	Y

20 ## Open Issues Log

21 Last updated: Aug. 6, 2010

Issue	Issue Date	Provided By	Summary of the Issue
50	8/5/10	Steve Van Ausdall	OpenADE SD - General-purpose authorization mechanism

22

Contents

List of Tables

1 INTRODUCTION

This document contains only the extensions necessary to the OpenADE Common specification to build a WS-I Basic Profile 1.1 implementation of the OpenADE Requirements Specification. The "OpenSG OpenADE SD – Common" document specifies the structure of the usage data payload transferred using the services defined in this specification.

These extensions define a collection of services, using SOAP over HTTPS to send and receive requests and information in XML. This architecture provides secure access to scalable methods and data resources hosted by the provider. All data is secured at the user level, so that access to individual user data 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

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 WEB SERVICE 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 the SOAP over HTTP environment, and possibly the web service definitions themselves.

1.2 REFERENCED SPECIFICATIONS

- [1] IEC CIM (TC 57 61968/61970) - http://tc57.iec.ch
- [2] OAuth - http://oauth.net/
- [3] WS-I Basic Profile Version 1.1 http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html
- [4] 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
- [5] IEC TC57 WG14 61968-1-2 – Profile for use of CIM with WS-I Basic Profile
- [6] 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
- [7] OpenADE SRS 1.0 - http://osgug.ucaiug.org/sgsystems/OpenADE/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fsgsystems%2fOpenADE%2fShared%20Documents%2fSRS

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] Service Definitions Technical Guide http://www.smartgridipedia.org/images/a/af/AMI_ENT_Step-By-Step_Modeling_and_Artifacts_Generation_Guidelines.doc
- [G3] UDDI: www.uddi.org

1.4 NAMESPACES

The subject of namespaces is important, because the namespace identifies the domain managing the definitions of protocol resources and formats. OpenSG proposes to use a temporary namespace until the final destination is identified. In any case, namespaces already defined elsewhere and used directly within reference service definitions will remain where they are, and will reference the identified body.

The proposed temporary namespace for definitions to be used in early implementations is below. (Service definition will be updated with the final approved namespaces.)

```
http://osgug.ucaiug.org/ns/2010/06/wsoade
```

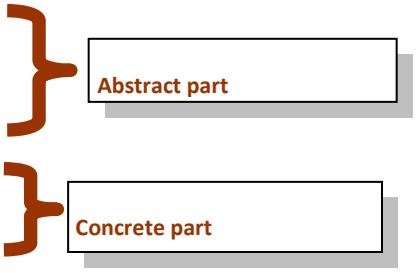
2 WEB SERVICES

The purpose of the section is to provide a set of recommendations for a Web Service definition based on OpenSG's Service Definition design patterns and Web Services Description Language (WSDL) from W3C. The audiences of the document are assumed to have basic knowledge of Web service and XML schema.

1.1 SERVICE STRUCTURE

W3C WSDL (v1.1) is followed to define OpenADE Web services. The services are made of two parts with following tags.

- **definitions**
- **types**
- **message**
- **portType**
- **operation**
- **binding**
- **port**
- **service**



The web service design practices are summarized below:

- Standard SOAP envelope is used to avoid extra message enveloping.
- XSD as data type is imported instead of being embedded for better version control
- Wire signature issue is avoided by redefining element names such as CreatedConsumption and ChangedConsumption using a single XSD Consumption complexType
- Wrapped document WSDL style is used
- Operation name follows the Verb + Noun naming convention which can potentially avoid contend-based routing

1.2 SERVICE NAMING CONVENTION

Interfaces are defined using a specific set of verbs and nouns using Web service technology. Each service then has a subset of operations that are associated with information objects. Each operation is named following IEC 61968-1 verb + noun (information objects). The detail service and operation naming convention is covered in OpenSG Service Definition Technical Guide [G2]. In summary, the naming conventions are:

- **Service name**:

  To follow ***<Service pattern name>+<Information Object>*** such as ReceiveConsumption

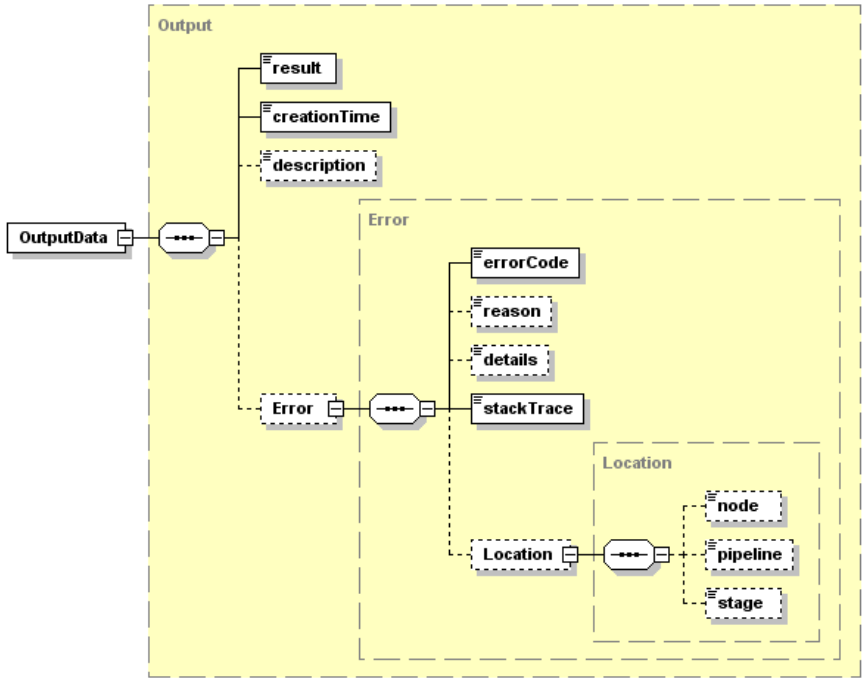
- **Operation name**:

  To follow ***<Operation pattern name>+<Information Object>*** such as CreatedConsumption

1.3 SOAP BINDING

The document style using SOAP body is the most common practice in WSDL design. It can fully utilize the benefits of an XML schema for payload validation.

Both <soap:binding> and <soap:operation> styles are defined as “document”. Also <soap:body> is used for both input and output operations. Input data type is typically a payload such as Consumption data definition. Output data follows a common XSD (OutputData.xsd) that is included for each operation in a WSDL. Each operation’s OutputData adheres to the following XSD structure and is used as an acknowledgement return or a fault return during a synchronous call.



The wsdl:*operation* is named the same as the input element name. As a result the WSDL is a wrapped document style WSDL. Wrapped document style originates from Microsoft to mimic a RPC style. In a RPC style, an XML payload is wrapped by its operation name.

Here is the WSDL section that illustrates the wrapped document style. Note the element name is the same as the operation name (CreatedConsumption):

```
 … …


<wsdl:message name="CreatedConsumptionInput">
    <wsdl:part name="CreatedConsumption" element="typeIn:CreatedConsumption"/>
</wsdl:message>


… …


<wsdl:portType name="Consumption">
    <wsdl:operation name="CreatedConsumption">
        <wsdl:documentation>CreatedConsumption</wsdl:documentation>
                <wsdl:input name="CreatedConsumptionInput" message="tns:CreatedConsumptionInput"/>
        <wsdl:output name="CreatedConsumptionOutput" message="tns:outputData"/>
        <wsdl:fault name="faultInfor" message="tns:faultReturn"/>
        </wsdl:operation>
</wsdl:portType>
```

167

168 One issue with the wrapped document style is when adding an “operation” like element in an XSD that may break
169 semantics in data definition. There can be also maintenance issue in a case of a new operation being added which
170 causes not only WSDL change but also XSD update. Therefore the recommendation is to create the operation like
171 elements within WSDL and decouple the original XSD element. Here is an example.

```
<wsdl:types>
 <xs:schema targetNamespace="http://<namespace prefix>/2009/09/ConsumptionIn">
     <xs:import namespace="http://<namespace prefix>/2009/09/Consumption" schemaLocation="Consumption.xsd"/>
     <xs:element name="CreatedConsumption" type="typeOrig:Consumption"/>
     <xs:element name="ChangedConsumption" type="typeOrig:Consumption"/>
  </xs:schema>
  … …
</wsdl:types>
```

Note that the operation-like element name is defined within wsdl:types section. This element references a complexType within Consumption.xsd which does not need a change for this style.

3 VERSIONING

Versioning will be handled in the manner specified in the OpenADE Common document.

Additionally, WSDL targetNamespace needs to be updated whenever a change occurs to an XSD namespace. In other words, a major XSD update will result in a WSDL namespace change and minor XSD update (no namespace change) will have no impact on WSDL namespace.

4 SERVICE OPERATIONS

The tables below list the service operations proposed in order to meet the requirements. These services will be fully specified in a subsequent publication.

4.1 PROVIDER (UTILITY) OPERATIONS

These operations are implemented by the provider of the data exchange service.

Operation	Inputs	Outputs	Description
GetServiceStatus	ResourceList	ServiceStatus	Synchronously check connectivity and current operational status of the service
RequestServiceStatus	ResourceList	RequestStatus	Asynchronously check connectivity and current operational status of the service
ReceiveServiceStatus	ServiceStatus	RequestStatus	Receive result of status check initiated by Utility
CreateEnrollment	Customer, Key, ResourceList	ActivityRecord	Initiate authorization of 3rd Party customer to receive Utility customer resources
CreatedEnrollment	Customer, ResourceList	ActivityRecord	Notify Utility of new authorization completion (future)
CancelEnrollment	Customer, ResourceList	ActivityRecord	Initiate cancel authorization of customer resources

Operation	Inputs	Outputs	Description
CancelledEnrollment	Customer, ResourceList	ActivityRecord	Notify Utility of authorization cancellation
GetActivityRecord	ID	ActivityRecord	Receive status of an asynchronous request from Utility
GetResource	Format	Resource	Transfer customer usage information data (or other resources, future)
ReceiveActivityRecord	ResourceList	RequestStatus	Notify Utility of current status of pending transfers

Table 1: Provider Service Operations

4.2 SERVICE CONSUMER (3RD PARTY) OPERATIONS

These operations are implemented by the consumer (client) of the data exchange service.

Operation	Inputs	Outputs	Description
GetServiceStatus	ResourceList	ServiceStatus	Synchronously check connectivity and current operational status of the service
RequestServiceStatus	ResourceList	RequestStatus	Asynchronously check connectivity and current operational status of the service
ReceiveServiceStatus	ServiceStatus	RequestStatus	Receive result of status check initiated by 3rd Party
CreateEnrollment	Customer, Key, ResourceList	ActivityRecord	Initiate authorization of Utility customer to receive 3rd Party customer resources (future)
CreatedEnrollment	Customer, ResourceList	ActivityRecord	Notify 3rd Party of new authorization completion (future)
CancelEnrollment	Customer, ResourceList	ActivityRecord	Initiate cancel authorization of customer resources
CancelledEnrollment	Customer, ResourceList	ActivityRecord	Notify 3rd Party of authorization cancellation
GetActivityRecord	ID	ActivityRecord	Receive status of an asynchronous request from 3rd Party
CreatedResource	ResourceList	RequestStatus	Notify 3rd Party that resources were created or updated
CreatedResource	ID	RequestStatus	Notify 3rd Party that new and updated resource files are available

Table 2: Consumer Service Operations

4.3 LARGE SIZE DATA EXCHANGE

It is recommended to use MTOM for large data transaction. MTOM stands for Message Transmission Optimization Mechanism. It is often used for a binary data transaction and usually used with XOP (XML-binary Optimized Packging). Using MTOM, the SOAP binding has no significant change in comparison with the conventional SOAP binding in document style. Currently there is no requirement on a large size payload data transaction. Should this be a case in the future, a new operation based on MTOM will be provided.

4.4 SERVICE DISCOVERY

Universal Description, Discovery, and Integration (UDDI) is a specification designed to allow businesses to enter details about themselves and the services they provide in a registry. Searches can be typically be performed by

199 company name, specific service, or types of service. This allows companies providing or needing web services to
200 discover each other, define how they interact over the Internet, and share information in a standardized fashion.

201 Since a WSDL defines the XML grammar for describing services as collections of communication endpoints capable
202 of exchanging messages, utilities and third parties can publish WSDLs for services they provide and links to the
203 WSDLs are usually offered in a company's profile in a UDDI registry.

204