Energy Interoperation Version 0.1

Working Draft

11 July 2009

Specification URIs:

This Version:

http://docs.oasis-open.org/energyinterop/energyinterop-wd-00_1.html
http://docs.oasis-open.org/enrgyinterop/ energyinterop-wd-00_1.doc
http://docs.oasis-open.org/enrgyinterop/ energyinterop-wd-00_1.pdf

Previous Version:

http://docs.oasis-open.org/enrgyinterop/ [additional path/filename] .html
http://docs.oasis-open.org/enrgyinterop/ [additional path/filename] .doc
http://docs.oasis-open.org/enrgyinterop/ [additional path/filename] .pdf

Latest Version:

http://docs.oasis-open.org/enrgyinterop/ [additional path/filename] .html
http://docs.oasis-open.org/enrgyinterop/ [additional path/filename] .doc
http://docs.oasis-open.org/enrgyinterop/ [additional path/filename] .pdf

Technical Committee:

OASIS Energy Interoperation TC

Chair(s):

David Holmberg
William T. Cox

Editor(s):

Toby Considine
[Editor name]

Related work:

This specification replaces or supersedes:

- [specifications replaced by this standard]
- [specifications replaced by this standard]

This specification is related to:

- [related specifications]
- [related specifications]

Declared XML Namespace(s):

[list namespaces here]
[list namespaces here]

Abstract:

Energy interoperation describes a data model and a communication model to enable collaborative and transactive use of energy. Web services definitions, service definitions consistent with the OASIS SOA Reference Model, and XML vocabularies for the interoperable and standard exchange of:

- Dynamic price signals
- Reliability signals

- Emergency signals
- Communication of market participation information such as bids
- Load predictability and generation information

This work facilitates enterprise interaction with energy markets, including but not limited to:

- Response to emergency and reliability events
- Take advantage of lower energy costs by deferring or accelerating usage
- Enable trading of curtailment and generation
- Support symmetry of interaction between providers and consumers of energy
- Provide for aggregation of provision, curtailment, and use

The definition of a price and of reliability information depends on the market context in which it exists. It is not in scope for this TC to define specifications for markets or for price and bid communication, but the TC will coordinate with others to ensure that commonly used market and pricing models are supported.

Status:

This document was last revised or approved by the Energy Interoperation Technical Committee on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/energyinterop/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/energyinterop/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/energyinterop/.

Notices

Table of Contents

1 Introduction

Energy markets have been characterized by poor coordination of supply and demand. This failing has exacerbated the problems caused by rising energy demand. In particular, poor communications concerning times of peak use cause economic loss to energy suppliers and consumers. There are today a limited number of high demand periods (roughly ten days a year, and only a portion of those days) when the failure to manage peak demand causes immense costs to the provider of energy; and, if the demand cannot be met, expensive degradations of service to the consumer of energy. As the proportion of alternative energies on the grid rises, and more energy comes from intermittent sources, the frequency and scale of these problems will increase. In addition, new electric loads such as electric vehicles will increase the need for electricity and with new load characteristics and timing.

Energy consumers can use a variety of technologies and strategies to shift energy use to times of lower demand and also to reduce use during peak periods. This shifting and reduction can reduce the need for new power plants, and transmission and distribution systems. These changes will reduce the overall costs of energy through greater economic efficiency. This process is called various names, including Demand Response (DR), demand shaping, and load shaping.

Distributed energy resources, including generation and storage, now challenge the traditional hierarchical relationship of supplier and consumer. Alternative and renewable energy sources may be placed closer to the end nodes of the grid. Wind and solar generation, as well as industrial co-generation, allow end nodes to sometimes be energy suppliers. Energy storage, for example mobile storage in plug- in hybrid vehicles, means that the same device may be sometimes a supplier, sometime a consumer. As these sources are all intermittent, they increase the challenge of coordinating supply and demand to maintain the reliability of the electric grid.

Better communication of energy prices addresses growing needs for lower-carbon, lower-energy buildings, net zero-energy systems, and supply-demand integration that take advantage of dynamic pricing. Local generation and local storage require that the consumer (in today's situation) make investments in technology and infrastructure including electric charging and thermal storage systems. Buildings and businesses and the power grid will benefit from automated and timely communication of energy pricing, capacity information, and other grid information.

Consistency of technology for interoperation and standardization of data communication can allow essentially the same model to work for homes, small businesses, commercial buildings, office parks, neighborhood grids, and industrial facilities, simplifying interoperation across the broad range of energy providers, distributors, and consumers, and reducing costs for implementation.

These communications will involve energy consumers, producers, transmission systems, and distribution systems. They must enable aggregation for production, consumption and curtailment resources. Market makers, such as Independent System Operators (ISOs), utilities, and other evolving mechanisms need to be supported so that interoperation can be maintained as the Smart Grid evolves. Beyond these interfaces, building and facility agents can make decisions on energy sale purchase and use that fit the goals and requirements of their home, business, or industrial facility.

The new symmetry of energy transactions demands symmetry of interface. A net consumer of energy may be a producer when the sun is shining, the wind is blowing, or a facility is producing co-generated energy. Each interface must support symmetry as well, with energy and economic transactions flowing each way.

This document defines the means of interaction between Smart Grids and their end nodes, including Smart Buildings and Facilities, Enterprises, Industry, Homes, and Vehicles. Interoperable dynamic pricing, reliability, and emergency signals are defined to meet business and energy needs, and scale, using a variety of communication technologies.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **RFC2119**.

1.2 Normative References

RFC2119 S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

RFC2246 T. Dierks, C. Allen *Transport Layer Security (TLS) Protocol Version 1.0*, http://www.ietf.org/rfc/rfc2246.txt, IETF RFC 2246, January 1999.

SOA-RM OASIS Standard, *Reference Model for Service Oriented Architecture 1.0*, October 2006. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

1.3 Non-Normative References

OpenADR v1.0 Piette, Mary Ann, Girish Ghatikar, Sila Kiliccote, Ed Koch, Dan Hennage, Peter Palensky, and Charles McParland. 2009. *Open Automated Demand Response Communications Specification (Version 1.0)*. California Energy Commission, PIER Program. CEC-500-2009-063.

BACnet/WS Web Services Interface ANSI/ASHRAE Addendum Cc to ANSI/ASHRAE Standard 135-2004.

SOA-RA OASIS Public Review Draft , *Reference Architecture for Service Oriented Architecture Version 1.0,* April 2008 http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf

1.4 Naming Conventions

This specification follows some naming conventions for artifacts defined by the specification, as follows:

- For the names of elements and the names of attributes within XSD files, the names follow the CamelCase convention, with all names starting with a lower case letter.
  eg <element name="componentType" type="energyinterop:ComponentType"/>
- For the names of types within XSD files, the names follow the CamelCase convention with all names starting with an upper case letter.
  eg. <complexType name="ComponentService">

- For the names of intents, the names follow the CamelCase convention, with all names starting with a lower case letter, EXCEPT for cases where the intent represents an established acronym, in which case the entire name is in upper case.
  An example of an intent which is an acronym is the "SOAP" intent.

1.4 Architectural References

Energy Interoperability defines a service oriented approach to energy interactions. Accordingly it assumes a certain amount of definitions of roles, names, and interaction patterns. This document relies heavily on roles and interactions as defined in the OASIS Standard *Reference Model for Service Oriented Architecture.*

2 Background

This work is based upon the Open Automated Demand Response Communications Specification (OpenADR) developed by Lawrence Berkeley Labs and Akuacom under a contract from the California Energy Commission. OpenADR defined the interfaces to the services provided by a Demand Response Automation Server (DRAS). A DRAS facilitates the automation of customer response to various Demand Response programs and dynamic pricing. OpenADR also addresses the interfaces used by third parties such as utilities, ISOs, aggregators (supply side) initiate DR communications and how energy and facility managers, and building automation vendors (supply side) will interface to the DRAS to automate response.

In this version, we limit Energy Interoperability services and interaction patterns between the supply and demand side, or roughly, to the outside of the DRAS server.

2.1 Purpose

The success of DR programs and dynamic pricing developed by utilities and ISOs depend upon timely and reliable communications of events and information to participants in the DR programs and dynamic pricing. If DR communications can be automatically translated into load sheds or shifts by the participants without the need for human intervention, then participating in demand response programs can be made more cost effective, reliable, and easy to implement.

Demand Response is a transitional approach as national energy markets move toward distributed energy resources (DER). Renewable energy sources such as wind and solar will provide intermittent energy increasing the need for rapid energy response. Energy storage, including customer-site-based storage will ameliorate the problems of intermittent energy. Widely deployed storage will complement site-base generation to change the relationships between energy suppliers and consumers. A likely outcome is a transition to transaction energy to keep energy markets always in balance.

2.2 Reason

Some participants such as aggregators and large corporations have wide spread geographical operations across multiple electrical jurisdictions and thus must deal with multiple utilities. Today, utilities and ISOs must perform systems integration and testing with each participant in a DR program. A standard interchange mechanism across multiple utilities and ISOs will reduce the effort and cost of participating in demand response programs and in dynamic pricing.

The acceptance and scope of DR relies on its acceptance by the customer. Direct utility control of customer equipments is necessarily *de minimis* and prone to defeat by gamin. Willing customer acceptance of DR is likely predicated on the ability of the customer opt in and out, and to control how to respond based upon his changing needs. Common communication standards across the country will create national markets for customer-based systems. A national market will create performance competition while it reduces costs by increasing scale.

2.3 Scope

Energy interoperability must support transactional energy as well as DR. Energy market operations are beyond the scope of this specification; the signals that manage the actual delivery and acceptance are within the scope.

3 Customer Interfaces Requirements

3.1.1 General Role of Demand Response Automation Server in Demand Response Programs and Dynamic Pricing

The DRAS is an infrastructure component in Energy Interoperability Automated Demand Response programs that facilitates the communications among the entities (e.g. utilities, ISOs) that produce and distribute electricity and the entities (e.g. facilities and aggregators) that manage the consumption of electricity.

Comment [TC1]: We need a name for the thingie so we can talk about it. DRAS is the old name, which made a lot of sense for OpenADR. Do we want to call it something else?

The purpose of the DRAS is to automate the various communication channels necessary for Automated Demand Response programs and dynamic pricing. Such communications include varied price and reliability related messages and information that are sent from utilities or ISOs to the various parties that manage the consumption of electricity in order to curtail the consumption of electricity during peak periods.

3.2 Use Cases

This section presents a typical use case of Automated Demand Response programs with the focus being the role of the DRAS in those programs and dynamic pricing. The use case presented in this section is a generalization. Appendix D contains use cases for specific DR programs and dynamic pricing, including detailed descriptions of the symbols and nomenclature used in the use case diagrams. The following roles are used in the use cases.

3.2.1 Use Case Scenarios

3.2.1.1 Utility-Based Roles

- **Utility Program Operator**. This is a human operator that manages various aspects of the utility's DR programs and dynamic pricing.

Comment [TC2]: Do we need this?

- **Program Notifier**. This is a computer sub-system or human operator that is responsible for notifying the participants of the DR events and related information.
- **Program Settlement**. This is a computer sub-system or human operator that is responsible for performing the settlements associated with DR programs and dynamic pricing by measuring the usage of electricity on a per participant basis and feeding the information into the utility's billing system.

Comment [TC3]: Do we need this?

3.2.1.2 DRAS Roles

- **Event Notifier**. This is a sub-system of the DRAS that notifies the participants about DR events initiated by the utility. This is specifically designed for the machine to machine communications necessary to automate the DR program.
- **RTP Notifier**. This is a sub-system of the DRAS that notifies the participants about real-time pricing (RTP) information as it becomes available. This is specifically designed for the machine to machine communications necessary to automate the DR program.

- **Program Notifier**. This is a sub-system of the DRAS that notifies participant operators of various events related to DR programs and dynamic pricing.
- **Bidding Proxy**. This is a sub-system of the DRAS that acts as an automated bidding proxy for DR programs and dynamic pricing that require participants to submit bids to the utility.

3.2.1.3 DRAS Client Roles

- **DRAS Event Client**. This is a sub-system of the DRAS Client and is responsible for notifying the facility's automation sub-systems about DR program events.
- **DRAS Feedback Client**. This is a sub-system of the DRAS that provides feedback to the DRAS concerning what is happening in a facility in response to a DR event.
- **DRAS Operator**. A human actor with the responsibility of creating other users.

3.2.1.4 Participant Roles

Participants are the customers of the utilities or ISOs that are participating in the DR programs and dynamic pricing. In general there will be one or more operators as part of the participant's organization that is responsible for managing various aspects of their involvement in the DR program. Within the context of the use cases, there are the following roles:

- **Facility Manager**. A human operator responsible for managing various aspects of the facility related to the DR program. Within the context of this document a facility manager may also be referred to as a "Participant Manager" or a "Participant Operator".
- **Aggregator Manager**. A human operator responsible for managing various aspects of the aggregator's participation in the DR program.

3.2.1.5 DRAS Performance Expectations

- The latency of DR events sent from the utility to the end user should be no more than 1 minute, depending upon the configuration of the interaction between the DRAS and DRAS Client.
- The DRAS must maintain accurate time within 15 seconds.
- The DRAS should have a means to allow participants to participate in multiple DR programs and dynamic pricing through the same DRAS.
- The DRAS should recover gracefully from facility faults with minimum lost data. Examples of such faults might be power failures or connectivity loss.

3.3 Use Case Scenarios

Each use case is presented with three broad scenarios:

- Program Configuration
- Program Execution
- Program Maintenance

Each of these scenarios discusses the actions taken by the various roles within that scenario.

3.4 Program Configuration

Program configuration consists of the actions taken to set up and automate a specific DR program with the emphasis being on the configuration of the DRAS to participate in the program. In some cases, configuration activities may be listed that are not related to the DRAS. These activities are listed only to give completeness to the overall DR program and will not be covered in detail.

3.4.1.1 Program Execution

Program execution consists of the actions taken to actually execute and participate in the Auto-DR program. This is the set of actions required for the utility to send DR-related information to the participants that are participating in the program. Emphasis is placed on the actions related to the DRAS. In some cases there may be activities listed that are not related to the DRAS. These activities are listed only to give completeness to the overall DR program and will not be covered in detail.

Comment [TC4]: This is very California specific. We need to discuss how much of this appears in the documents

3.4.2 Generalized Use Cases for Demand Response Programs

Table 2 is a spreadsheet that shows the various demand response functions (actions) as they exist across the various use cases documented in Appendix D and is useful for identifying actions which are common across all programs and dynamic pricing. Note that the actions are generalizations of the various actions across all the programs and dynamic pricing listed above.

The following generalizations can be made with respect to the operation of the DRAS:

- The use cases which include the use of aggregators are very similar to the use cases which operate directly with facilities and facility managers; therefore it is reasonable to treat the aggregator roles and the facility manager roles to be equivalent.
- All the use cases that are only for propagating events have very similar sets of steps for the various scenarios. The differences are related to the type of information transmitted with the events.
- All the use cases that automate the bidding process include a very similar set of steps.
- Although the bidding process is linked to specific events there is not a strong coupling between the steps used in the bidding process and the steps involved in propagating events.

In analyzing the DR programs and dynamic pricing so far one can see that there are two general classes of functions:

- Actions related to the automation of DR event notification.
- Actions related to the automation of the DR bidding process.

Based upon this analysis, it is possible to generate general use cases that cover these two functional classes. These use cases are presented in the subsequent sections.

DR Program Execution

Program		Configuration			DR Program Execution								Maintenance	
					Actions on DRAS					Actions By DRAS				
		Configure Program	Configure DRAS Client Connection	Configure Bids	Request Bids	Initiate DR Event	Program Opt Out	Set Load Status	Set Current Bids	Send DR Event Info	Notify Request for Bid	Notify Bid Status	Utility Operator Reports	Client Reports
CPP	Utility Operator	X											X	
	Utility Program Notifier					X								
	Utility Info System													
	DRAS Client							X		X				
	Client Operator		X				X							X
DBP	Utility Operator	X											X	
	Utility Program Notifier				X	X								
	Utility Info System								X					
	DRAS Client							X		X				
	Client Operator		X	X			X				X	X		X
CBP	Utility Operator	X											X	
	Utility Program Notifier				X	X								
	Utility Info System								X					
	DRAS Client							X		X				
	Client Operator		X	X			X				X	X		X

Program					DR Program Execution									
		Configuration			Actions on DRAS					Actions By DRAS			Maintenance	
		Configure Program	Configure DRAS Client Connection	Configure Bids	Request Bids	Initiate DR Event	Program Opt Out	Set Load Status	Set Current Bids	Send DR Event Info	Notify Request for Bid	Notify Bid Status	Utility Operator Reports	Client Reports
BIP	Utility Operator	X											X	
	Utility Program Notifier					X								
	Utility Info System													
	DRAS Client							X		X				
	Client Operator		X				X							X
PDC	Utility Operator	X											X	
	Utility Program Notifier					X								
	Utility Info System													
	DRAS Client							X		X				
	Client Operator		X				X							X
PCT	Utility Operator	X											X	
	Utility Program Notifier					X								
	Utility Info System													
	DRAS Client							X		X				
	Client Operator		X				X							X

Program					DR Program Execution									
		Configuration			Actions on DRAS					Actions By DRAS			Maintenance	
		Configure Program	Configure DRAS Client Connection	Configure Bids	Request Bids	Initiate DR Event	Program Opt Out	Set Load Status	Set Current Bids	Send DR Event Info	Notify Request for Bid	Notify Bid Status	Utility Operator Reports	Client Reports
RTP	Utility Operator	X											X	
	Utility Program Notifier					X								
	Utility Info System													
	DRAS Client							X		X				
	Client Operator		X				X							X

4 Energy Markets and Considerations

4.1 Ed Cazalet's comments

I would like to get Ed's comments here in the focus and purpose part of this documents.

Day Ahead Pricing

Hours Ahead Pricing

Five minute pricing

5 Interaction Patterns

5.1 Generic Bidding Process

The bidding process includes the following steps:

1. The utility program operator creates a GBP bidding event in Utility Information System. In this step, a program operator schedules a GBP bidding event in the Utility Information System. The details of this process are beyond the scope of this document.
2. The utility program notifier gets GBP bidding event information from the Utility Information System and initiates GBP Request for bids in the DRAS. The information sent to the DRAS by the utility program notifier sub-system includes the following information:
   - Program type
   - Date and time of the event
   - Date and time issued
   - Geographic location
   - Participant list (account numbers)
   - Request For Bids (RFB) issue date and time
   - RFB close time
   - Price offered for load reduction per time block
3. The DRAS program notifier sends request for bid to the participant manager. This notification typically comes in the form of an email, phone call or page.
4. The participant manager can adjust or cancel the current bid in the DRAS. This is an optional step and allows the manager to adjust their bid for that particular event. If this step is not performed then the DRAS will submit the standing bid after the end of the bid period.
5. After the bidding time limit has expired, the bidding proxy in the DRAS sets the current bid in the UIS. The information sent by the DRAS includes the following for each participant:
   - Participant account number
   - Load reduction bids per time block
6. The utility program notifier gets the accepted bids from the Utility Information System and sets the accepted and rejected bids in the DRAS. The information concerning the accepted bid includes the following:
   - Participant list (account number)
   - Accept or Reject
   - Load reduction bids per time block (for verification)
7. The DRAS program notifier sends the acceptance or rejection notification to the participant manager via phone, email, and/or page.

5.2 Simple vs. Smart DRAS Clients

The DRAS supports two different types of DRAS Clients – Simple and Smart.

The Smart DRAS Client is assumed to be capable of dealing with all the *EventInfo* information that may be associated with a DR event that is initiated by the utility or ISO. It can parse all the *EventInfo* information and make decisions about how to respond to that particular DR event information.

On the other hand there are many cases where a Simple DRAS client is needed. These cases have simplified EMCS that are incapable of any sophisticated logic or the ability to deal with the wide range of information types that may be associated with a DR event. In these cases, the DRAS translates the *EventInfo* information associated with a DR event into a much simpler form, known as a Simple DRAS Client.

The DRAS must be capable of dealing with both Smart and Simple DRAS Clients. Details on both of these scenarios are described below.

5.2.1 DR Event Information

DR programs are typically designed to use a variety of information to cause reactions by participants to DR events that are issued by the utility or ISO. In some cases prices are used to trigger responses to the DR events while in other cases it might be a shed or shift level. In general there can be a wide range of different types of information associated with a DR event depending upon how the DR program was designed. Therefore, the data models used to describe the information associated with DR events are designed to accommodate the wide range of information that may be associated with a DR event. This information is represented by *EventInfoInstance* entities.

When a program is defined within the DRAS there are specifications associated with the program that define what type of information may be associated with a DR event when one is issued for that program. Each type specification for an *EventInfoInstance* is referred to as an *EventInfoType*. A program may be defined that allows for multiple different types to be associated with a program. Each *EventInfoType* contains the following attributes and elements:

EventInfoType

- **Name** – This is the name of the type. Analogous to a variable name.
- **typeID** – This identifies the type of information and may take on one of the following values:
    - PRICE_ABSOLUTE - Price number, i.e. $0.25
    - PRICE_RELATIVE - Change in price, i.e. -$0.05
    - PRICE_MULTIPLE - Multiple of current price, i.e. 1.5
    - LOAD_LEVEL - Amount of load based on an enumeration, i.e. moderate, high, etc.
    - LOAD_AMOUNT - Fixed amount of load to shed or shift, i.e. 5 MW
    - LOAD_PERCENTAGE - Percentage of load to shed or shift, i.e. 10%
    - GRID_RELIABILITY - Number signifying the reliability of the grid

- **scheduleType** – This specifies how a schedule may be associated with the DR Event information is defined and may take on the following values:
    - **NONE** – There is no schedule and thus the *EventInfo* does not change values during the entire DR event ACTIVE state.
    - **DYNAMIC** – The time schedule is not fixed during configuration, but can be set when the DR event is issued.
    - **STATIC** – The schedule is fixed when the DR program is configured within the DRAS
- **Schedule** – If the *scheduleType* is STATIC then this is the configured schedule. A schedule is a sequence of time slots that are valid over the entire ACTIVE period of a DR event. Each time slot may take on a different value in the *EventInfoInstance*
- **enumerations** – This is a list that defines a fixed set of values that the *EventInfo* instance may take. If defined, the *EventInfoInstance* is an enumeration and can take on any of the values in the list. If left undefined, the *EventInfo* instance can take on any contiguous value between the *minValue* and *maxValue*.
- **minValue** – Minimum possible value of an *EventInfoInstance*.
- **maxValue** – Maximum possible value of an *EventInfoInstance*.

Note that when a DR event is issued, the *EventInfoInstance* variables that are associated with the DR event may take on values that change according to some schedule during the ACTIVE state of the DR event. Also note that the schedule that defines when these values change may be defined as part of the definition of an *EventInfoType* or it may be defined when the DR event is issued. See Section **Error! Reference source not found.** for a detailed description of the *EventInfoType* schema.

Note that the DRAS Client never sees the *EventInfoType* definitions within the DRAS. Nonetheless it is useful to understand their structure in order to understand how the DRAS translates information from the various *EventInfoInstance* variables described below into the simple levels used by a Simple DRAS Client.

The *EventInfoType* defines the type of information that is associated with a DR event and are specified as part of a program. Thus when a DR event is actually initiated it contains instances of the *EventInfoType* that were defined to belong to the program. These instances are referred to as an *EventInfoInstance* and from the DRAS Clients point of view are defined as follows.

<u>*EventInfoInstance*</u>

- *eventInfoName* – This is the same as the name field in the corresponding *EventInfoType* definition.
- *eventInfoTypeID* – This is the type identifier of the data and takes on the value defined in the *typeID* of the *EventInfoType* definition.
- *eventInfoValues* – These are the actual values of the instances. There may be more than one if there is a schedule of values.

See Section **Error! Reference source not found.** for a detailed description of the *EventInfoInstance* schema, especially as it applies to how the schedule of values is defined.

5.2.2 Smart DRAS Client Event Information

The *EventState* message sent to a DRAS Client contains the following set of fields:

- Smart DRAS Client type data–this information is intended to be used by Smart DRAS Client
  - Event notification time
  - Event start time
  - Event end time
  - Event information (*EventInfoInstance*)

This information is derived directly from the DR event (*UtilityDREvent*) that was issued by the utility or ISO. The "event information" field is simply the collection of *EventInfoInstance* data as described above. In general the event timing (notification, start, and end times) is the same as what was specified in the DR event when it was issued by the utility or ISO, but there may be exceptions if there were *ProgramConstraint* variables defined within the DRAS that caused these values to be altered. For more information on *ProgramConstraint* see Section **Error! Reference source not found.**.

It is assumed that a Smart DRAS Client is capable of parsing and dealing with the various fields defined above.

5.2.3 Simple DRAS Client Event Information

It is assumed that a Simple DRAS Client type is not capable of dealing with the sophistication of the information that a Smart DRAS Client is. Therefore, the *EventState* information for a participant with a Simple DRAS Client is a simplified derivation of the information sent by the utility or ISO when the DR event is initiated. In this case there are the following two state variables that describe the DR event state:

- **Operation Mode:** This depicts the operational state of the facility and can take on the following values:
  (1) NORMAL operation
  (2) MODERATE shed or shift
  (3) HIGH shed or shift
  (4) SPECIAL
- **Event Status:** This depicts the current temporal state of a DR event and can take on the following values:
  (1) NONE – no event pending
  (2) FAR – event pending far into the future.
  (3) NEAR – event pending soon.
  (4) NOW – event currently in process.

In general, the Event status variable always transitions from NONE to FAR to NEAR to NOW. The transition from FAR to NEAR is a configurable parameter of a program and in fact there may not even be a NEAR state in which case the FAR value could simply be interpreted as meaning “Event Pending”.

The Operation Mode variable takes on values according to a schedule during the event that is defined by the participant or the utility or ISO. This schedule is specified by using a set of rules that determine how the *EventInfoInstance* of the *UtilityDREvent* is translated into one of the simple values of the operation mode. Since the participant is free to schedule how the Operation Mode variable changes, this defines a so called “Response Schedule” for how that participant responds to DR events. The response schedule is represented by the *ResponseSchedule* entity.

A *ResponseSchedule* is an ordered list of rules represented by the *OperationStateSpec* entity. An *OperationStateSpec* within a *ResponseSchedule* represents a set of rules that are valid within a specific time slot of the ACTIVE period of the DR event. These rules dictate how the Operation Mode variable will transition during the time slot of the *OperationStateSpec.* The time slots that define the different *OperationStateSpec* entities form points at which the Operation Mode values may transition, but the Operation Mode values may also transition at times in the middle of the time slot associated with an *OperationStateSpec* if the value associated with an *EventInfoInstance* happens to change then.

The rules within an *OperationStateSpec* are expressed in terms of a table wherein each row in the table represents a Boolean equation such that if the equation is true then the corresponding Operation Mode value will be set. The equations are Boolean comparisons of the existing *EventInfoType* names for this program. Table 3 is an example table for a single *OperationStateSpec* entity where there are two *EventInfoTypes* with the names of RTP and BID that were defined for the program.

Table 1 OperationStateSpec Entity

Value	Equation
MODERATE SHED OR SHIFT	RTP > 5 & BID > 10
HIGH SHED OR SHIFT	RTP > 10 & BID > 10
MODERATE SHED OR SHIFT	RTP > 5 & BID < 5
SPECIAL	RTP > 15
NORMAL OPERATION	TRUE

Source: Lawrence Berkeley National Laboratory/ Akuacom

Note that each row in the table is evaluated from top to bottom until one of the equations is true. Whichever equation is true then the corresponding value is used to set the operation mode value. Note that if none of the rows are true then the operation mode value does not change. It is therefore good practice to put a default TRUE value at the very end which will be used if none of the other equations are true.

The following Boolean operations should be supported:

- AND
- OR
- XOR
- NOT
- GREATER THAN, GREATER THAN OR EQUAL

- LESS THAN, LESS THAN OR EQUAL
- EQUAL, NOT EQUAL
- GROUPING, i.e. parenthesis

In general the equation can be represented as a simple string. It is beyond the scope of this document to define the exact syntax of the rules strings.

Comment [TC5]: Clearly, if we include this, we should define it within the OASIS process

5.2.4 DRAS Client State

From the DRAS point of view a DRAS Client may be in one of the following modes of operation:

- **opt-in** – DR events are handled and sent to the DRAS Client as they normally would be.
- **opt-out** – The DRAS Client has opted out from receiving DR event information and none will be sent when they are in this state.
- **Test** – This is used for test purposes and is analogous to the DRAS Client being off line. No DR events will be sent automatically from the DRAS, but a DRAS installer may send test messages to the DRAS Client.

Note that the states listed above are all from the point of view of the DRAS and the DRAS Client responds to the state the DRAS is in. It is the DRAS that changes its behavior and where the logic resides in relation to the DRAS Client.

The DRAS is responsible for tracking the event states for each of the DRAS Clients in order to send the DR event information to the DRAS Client at the appropriate time. From the DRAS Client's point of view there is a so-called DR event state the DRAS Clients are in which is represented by the *EventState* entity. Normally a DRAS Client's event state is "IDLE" meaning that there are currently no active or pending DR events. This changes when the utility or ISO initiates a DR event in the DRAS. The DRAS tracks the DR event state for each DRAS Client and can provide the current state information at any time for that DRAS Client. It can be in different states, depending upon whether the participant uses a Smart DRAS Client or a Simple DRAS Client.

5.3 Automated Bidding Models

The DRAS may support automated bidding by participants into DR programs by supporting a "standing bid" for that participant. A standing bid is a bid that will be submitted by the DRAS for a participant if no other bid is submitted by the participant. The ability to automatically submit standing bids increases the level of participation in programs that require bidding. \

There may be scenarios where there are programs that require bidding, but all the bidding is handled by a different system than the DRAS, including the handling of standing bids. In this case, from the DRAS point of view, the program will not require bidding and all the DR events are simply issued by the utility or ISO as in the case of programs with no bidding. The utility or ISO will simply handle all the bidding as they normally would and use the DRAS to issue the DR events.

Comment [TC6]: Thesedefinitions should be alligned with the standard marketdefinitions that appear in the FIX protocol

Comment [TC7]: Does this al go away as we develop the EMIX suite?

The remainder of this section assumes that the DRAS is handling the automated submission of standing bids. From the participant's point of view there is a so called bid state that they are in (see Figure 37). Normally the state is "IDLE" meaning that there are currently no outstanding requests for bids. This changes when the utility or ISO initiates a DR event for a program that may require bidding. In this case the DRAS will issue a request for bids by notifying the participant operators via email or some other means. The DRAS then tracks the bid state for each participant and can provide the current state information at any time for that participant. The state variables associated with each bid request include:

- **Program** – the program associated with requests for Bids.
- **Notify time** – the time that participants are notified of the request for Bids.
- **Start time** – the start time of the bidding
- **End time** – the end time of the bidding
- **Bid info** – this gives program and DR event related information related to bidding.

The *Bid* entity is used by both the utility or ISO and the participants to represent bids as described in Section **Error! Reference source not found.**. Bid state transition diagrams are shown in Figures 38 and 39.

There is a separate state diagram for each program and DR event that a participant may submit bids for. The bidding sequences in Figure 38 are initiated by the utility or ISO when they initiate a DR event for a program that requires bidding by the participant. In some cases the bidding is open and closed according to some fixed schedule which is not associated with a specific event. In this case the opening and closing of the entry of bids is simply according to some schedule.

When the utility or ISO notifies the DRAS of all bid acceptances (step 5), the acceptances come in the form of a list of *UtilityDREvent* entities that describe each of the DR events that need to be issued to the participants whose bids were accepted. This allows for the utility or ISO to customize a DR event for each participant that will reflect the bids that they made. Note that it is a requirement that each of the *UtilityDREvent* entities that are issued as part of the list of accepted participants have the SAME event identifier as the original DR event that was issued and initiated the request for bids.

Upon receiving the list of *UtilityDREvent* that signify the participants whose bids were accepted the DRAS will notify the participant operators (Step 6) and send the DR events to the relevant DRAS Clients (Step 7).

5.4

5.5 Verification and Compliance

As the Energy Interoperation signals exist in two entirely different business models, Demand Response and Transaction Energy,twodifferent models are r

5.5.1 Demand Response Verification

Area needs considerable discussion

Demand Response is an odd beast. It is focused on change, not performance. The thinking is: we can't really prove what energy you would or would not have used, so instead we will focus on did you perform specific acts irrespective of whether or not any particular amount of energy is used.

I understand it, but we will need to discuss it for some time to select the optimum forward-looking description.

done,

6 Energy Interoperability semantics

6.1 Supply-side perspective on DR Events

When the utility or ISO initiates a DR event it uses an *UtilityDREvent* entity which contains the following general attributes:

- The DR program the event is for.
- Time and date parameters concerning when the event will take place.
- Time and date parameters concerning when participants should be notified of the upcoming event.
- Who and/or where to send the DR event information.
- Information associated with the event (*EventInfo*). This is program specific information that is related to the event, e.g. RTP or shed or shift level.

6.2 Supply-side semantic elements

These are the entities used when the Utility or ISO initiates a DR event. In particular the *UtilityDREvent* entity is used to specify all the information associated with a DR event and contains the following general attributes:

- **eventIdentifier** – This is a globally unique ID that is specified by the utility or ISO when the DR event is issued. It is subsequently used to associate and retrieve information related to a particular DR event.
- **programName** – This is an identifier that specifies which DR program the DR event is being issued for.
- **eventModNumber** – This is a sub identifier to the event identifier and is used specifically to determine when changes have been made to the DR event information since the last time it was issued. For example, the very first time a DR event is initiated by the utility or ISO, this has a number of 0 and will continue to have a value of 0 for each and every subsequent transmission unless the DR event is subsequently modified by the utility or ISO. At that time this number is increased to the next version number, indicating the original event was modified.
- **utilityITName** – This is an optional field which is the name and/or version number of the utility IT system that initiated the DR event.
- **Destinations** – This is a list of identifiers that specifies who is to receive the DR event. Note that it can be any of the following:
    - Explicit Participant Account User ID(s) (uid)
    - Group identifiers
    - DRAS Client location specifications
- **eventTiming** – Various timing parameters for the event, including:
    - **notificationTime** – This is the time at which the participants should be notified of the DR event.
    - **startTime** – This is the date and time that the DR event becomes active.
    - **endTime** – This is the date and time that the DR event ends.

- **biddingInformation** – If a program supports bidding by the participants then the following fields are also included.
  - **openingTime** – This is the time at which the participants may start placing bids.
  - **closingTime** – This is the time at which the bidding will close and is the deadline for which the utility may receive the bids from the DRAS.
- **Event Information** – This is the information associated with the DR event and is a list of *EventInfoInstance* entity as described below.

6.2.1 Supply-side sub-schema

Each *UtilityDREvent* may contain a list of *EventInfoInstance* variables which are used to describe the information that may accompany a DR event. Each *EventInfoInstance* entity contains the following general attributes:

- **eventInfoTypeName** – this is a name that is used to correlate this value against a predefined type of values for this program as described in Section 5.2.3. Note that the type name provides a way for the various values associated with a DR event to be identified. The various types are defined when the DR program is defined and given names which are referred to here.
- **Values** – these are a list of one or more values of the defined type. If there is more than one value then each value corresponds to a specific time slot within the DR event ACTIVE period. Essentially they will form a schedule of values. This is described in more detail in Section **Error! Reference source not found.**.
- **Participants** – this is a list of participants for which these values apply. It allows the information associated with a DR event to be applied to specific participants.
- **Groups** – this is a list of groups for which these values apply. Since participants may belong to a group, groups are another way of specifying which participants are to receive the information.

As can be seen by the diagram it is assumed that the following entities exist before an *UtilityDREvent* is created that refer to them:

- *UtilityProgram*
- *ParticipantAccount*
- *EventInfoType*

6.2.2 Consumer Opt-Out

A participant may choose to opt-out or override participating in DR events. The participant does this by using the *OptOutState* entity to set up one or more conditions within the DRAS that define when the participant will not participate in a DR event. The *OptOutState* may have the following attributes:

Comment [TC8]: IT seems likely that we need something derived from this section to handle the many programs and opt-in vs opt out DR

- Which DR programs the conditions apply to. If not specified, it defaults to all programs.
- Which DRAS Clients the conditions apply to. If not specified, it defaults to all DRAS Clients.
- Which DR event the conditions apply to. If not specified, it defaults to all DR events.
- A schedule which defines when the conditions apply.

Note that there can be more than one such set of conditions established by the participant. See Section **Error! Reference source not found.**.

Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here.

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

[Participant Name, Affiliation | Individual Member]
[Participant Name, Affiliation | Individual Member]

B. Non-Normative Text

C. Revision History

Revision	Date	Editor	Changes Made
0.1	[Rev Date]	Toby Considine	Initial transcription from OpenADR work donated by the California Energy Commission and Lawrence Berkeley National Labs