CALCONNECT THE CALENDARING & SCHEDULING CONSORTIUM

CALCONNECT DOCUMENT CD1004

Type:	**Proposal**
Title:	**Cal-WS Web Services API for Calendaring and Scheduling**
Version:	**0.3**
Date:	**2010-05-19**
Status:	**Public Review**
Source:	**XML Technical Committee**

This Proposal is an in-progress working document which has been made available for a 30-day public review and comment period. Please offer any comments or suggestions to the CalConnect Public Review and Comment Mailing list. For information about this list and to subscribe please see

http://www.calconnect.org/publicreviewdocuments.shtml.

Cal-WS Draft V0.3

May 19, 2010

Define a web service that allows basic CRUD (create, read, update, delete) operations, query, and scheduling operations on a calendar and meets the requirements of uses cases related to the Smart Grid initiative.

Smart Grid Suppliers and Consumers

Smart grid scenarios involve two main actors – a supplier and a consumer.

Goals

Assumption: A smart grid consumer wants to schedule consumption of grid capacity to meet needs while minimizing cost or total energy use.

Assumption: A smart grid supplier wants to schedule the available supply of grid capacity to meet consumer demand while minimizing outages and unused capacity.

To achieve the goals above, consumers and suppliers must be able to do different but related activities:

Consumer	Supplier

Anticipate demand	Communicate future anticipated demand.	Determine how much capacity consumers will demand at some point in the future.
Anticipate capacity	Determine available capacity and its cost now and in the future.	Communicate available capacity and its cost now or in the future.
Sell capacity	Purchase a portion of grid capacity from a supplier (or another consumer) in advance.	Sell a portion of grid capacity in advance.
Re-sell or buy back capacity	Sell back, re-sell, or give up rights to grid capacity.	Buy back grid capacity from consumers.
Avoid outages	Gracefully adjust when less capacity is available than it needs.	Force consumers at times to use less capacity to avoid outages.

Simple Interaction Model

For the purposes of this proposal, here is a simple model on the interactions between consumers and suppliers:

Interaction	Initiated by	Targeted to	Purpose
Ask	Consumer	Supplier	Communicate demand and request an offer.
Offer/Bid	Supplier (or consumer)	Consumer (or supplier)	Offer available capacity for sale (or re-sale) and its cost for purchase.
Purchase/Reject	Consumer (or supplier)	Supplier (or consumer)	Decision to accept or reject the offer.
Restriction	Supplier	Consumer	Signal consumer to reduce consumption to avoid outage.

Use Cases

Publish and query a rate schedule

A utility defines a rate schedule as a collection of “events” which include utility rates for a defined period of time. For example, different electricity rates can be set for peak and non-peak hours. The utility creates events in the rate schedule and makes it available to be queried by devices on the grid.

Ask for Bids (negowatts)

Consumer will reduce its demand by not using power it has already purchased if it receives an acceptable offer.

- Offers can be made between 3 and 4 this afternoon (event)
- Demand will be reduced between 4 and 4:30 p.m. this afternoon (event)

Ask for Bids (load profile)

Consumer wishes to buy power on a recurring basis with a specific usage profile.

- Offers can be made between 3 and 4 this afternoon (event)

- Load profile of several consecutive 30 minute intervals, each requiring different power (sequence)
- Recurring schedule could be one of the following (recurring event)
    - Weekday evenings at 6 p.m. starting May 1 for 6 weeks
    - Weekday mornings at 4 a.m. starting April 15 for 6 weeks
    - Saturday mornings at 8 a.m. starting April 1 for 15 weeks

Submit Bid

Supplier agrees to supply power under certain conditions.

- Consumer must reply by Friday at 4:30 p.m. (datetime)
- Supplier can provide power between 12 a.m. and 10 a.m. every day this week (recurring event)
- Supplier can respond within half a second (duration) for up to 20 minutes (duration)

Other Protocols

In developing this protocol attention has been paid to protocols that already exist in the calendaring arena. Most appear to support a similar conceptual model but there are a number of detail differences.

CalDAV

CalDAV appears to be closely aligned to the requirements for this protocol:

- It is XML based
- It is designed specifically for client/server calendaring interactions
- It can transport different content-types.
- Handles CRUD, reporting and scheduling operations.

CalDAV as it stands is not appropriate for use as a web service as it extends WebDAV and uses that protocol to provide authorization and access control.

However, the experience of designing and implementing CalDAV provides us with some important experience which can be applied to this protocol.

CalDAV data structures

CalDAV, building upon WebDAV, assumes a hierarchical structure of collections (corresponding to file-system folders) and resources (corresponding to file-system files). It applies some further constraints to that structure, certain collections are defined as calendar-collections and these can only contain calendar resources, events, tasks etc.

Resources and collections are targeted by a path structure allowing simple http GET and PUT operations to access those resources.

CalDAV CRUD

The basic CRUD operations in CalDAV are provided by the http PUT, GET and DELETE methods. The semantics are essentially the same as those for the basic http methods.

CalDAV Collection operations

CalDAV collections are created by the WebDAV MKCOL method which now allows the addition of a body to provide properties for the new collection. They are deleted by using the DELETE method. Any contained resources and collections are deleted along with the collection.

WebDAV and CalDAV do not define the semantics of GET on a collection. Some CalDAV servers will deliver a complete iCalendar VCALENDAR component containing all resources, others will return html allowing browsing of the structure.

CalDAV reports

CalDAV reports build upon the WebDAV report method and provide some basic querys and filtering. Reports can take some different forms:

- Query and filter
    - Query matches for a number of constraints
    - Filter specifies what properties to return
- Multi-get
    - Specifies a number of target objects to return
- Freebusy report
    - This returns freebusy information for the targetted resource. In general it is NOT the same as the freebusy for a principal.

CalDav Scheduling

From the client perspective scheduling is very simple. In essence, a calendar is considered the scheduling calendar collection and events placed in that collection which can be considered meetings (they have attendees) are automatically sent to the attendees. As responses come back to the organizer the meeting in the calendar is automatically updated and new copies broadcast to the attendees.

An inbox is defined to hold the incoming messages and these messages act as notifications to the client that something has changed.

The messages sent conform in all cases to the iTip specification - RFC5546.

Scheduling freebusy queries.

These are obtained through a POST to the targeted principals outbox. The response is an iTip representation of the principals freebusy status wrapped up in an xml body which also contains status information.

Note that this freebusy request is NOT targeted at any specific resource but rather at the recipient. The server is free to use any information that is appropriate to build a response, including workday information, external resources and so on.

CalDAV Access Control

Access control is based on WebDAV Acl and currently requires that clients manipulate access control lists (ACLs) to set desired levels of access. CalDAV has extended the access rights to include a significant number of scheduling rights.

CalDAV synchronization

The support for synchronization capabilities is a work in progress. WebDAV relies on Etags which have proved inadequate. A ctag has been defined as a vendor extension, supported by most servers, which allows clients to determine that a collection has been changed. Work is in progress on further DAV extensions to help synchronization.

Implications of other protocols for CalWS

Structure:

The DAV like hierarchical structure would seem appropriate. It allows references to entities and collections through a path which uniquely identifies each node.

Reports:

The ability to limit data by ranges and by properties in vital. Interoperability might be enhanced by the definition of an abstract calendaring query language.

Scheduling:

The CalDAV implicit scheduling approach has many benefits, in particular, simplicity.

Synchronization:

Synchronization of calendars, or at least a part of a calendar, is of particular significance. Synchronization methods need to limit the amount of data that needs to be transferred.

Access Control

The ACL approach to access control has proved to be a problem both in implementation and use. It is confusing to the end user and most user interfaces simply don't support it. An approach such as intentional access control (e.g. "make my calendar readable by that group") allows servers to implement the sharing in any way they wish without needing to reveal acls.

Capabilities

It would be approrpriate to build in some form of capabilities report from the start. This allows either end of a conversation to indicate what they support.

Representing Calendar Data

iCalendar in XML (xCal)

Assumption: Whenever calendar items are specified in CalWS, the iCalendar in XML (xCal) format will be used.

xCal defines the representation of common calendar concepts such as dates, datetimes, durations, and single or recurring events.

Extensions to xCal

iCalendar was originally formulated to deal with interpersonal scheduling and as a result does not define some concepts that are relevant to smart-grid scheduling scenarios.

Assumption: In cases where xCal does not define important smart grid scheduling concepts, an XML extension will be defined in a CalWS namespace and mapped to an x-property or x-param in iCalendar.

Precision

Precision is defined as the number of significant digits used to express date times or durations.

Issue: The minimum precision required for smart grid datetimes must be milliseconds, while ICalendar currently only supports seconds.

There is no way to extend the iCalendar value type DATE-TIME without breaking compatibility.

Proposed: Use a TimeFraction Perhaps a FLOAT value specified in xCal for properties with date time or duration values:

```
<x-TimeFraction>0.1245</x-TimeFraction>
```

Uncertainty

Uncertainty is defined as a "plus or minus" value. Example: At such-and-such a time for so long, beginning (or ending) within "x" time units of the specified start (or end) time.

Issue: ICalendar only deals with exact times. Uncertainties can be applied to a start time or end time.

Proposed: Use a DURATION value specified in xCal for in a new Uncertainty x-property with date time values:

```
<x-StartTimeUncertainty>P3H20M</x-StartTimeUncertainty>
```

Uncertainty needs to be expressed with a precision of milliseconds.

Calendar

A calendar is defined as a set of related events with a common identifier. ICalendar defines events but it doesn't attempt to define *calendars*. For example, it doesn't define a name for the calendar or an id that be used to distinguish one calendar from another.

Isn't there a proposal to add a calendar identifier to iCalendar?

Sequence

An ordered set of durations with specified offsets without a definite start time. Example: Profile of power usage in 30 minute intervals.

Proposed: Use of a set of related VTODOs to specify a sequence of durations. Introduce a new reltype param to indicate the relation of the VTODOs to each other.

Response time

An action must be taken within some time defined by an event. Example: making or responding to an offer by some deadline. Example: delivering grid capacity within some duration after an agreed-upon start time.

Constrained event

Constrained event is defined as an event that occurs within some time range. Example: 20 minutes of power supplied sometime between midnight and 8 a.m.

Scheduling Protocol and Methods

Assumption: If smart grid scenarios require scheduling, ITIP will be taken as the starting point for CalWS scheduling methods.

Question: Do smart grid use cases between consumers and suppliers really require scheduling methods, or just flexible descriptions of time?

Several ITIP methods are involved in interpersonal scheduling – REQUEST, REPLY, COUNTER, CANCEL. These methods don't seem suited to the smart grid use cases that would require

- A set of events to be scheduled on an all-or-nothing basis.
- One of several alternative recurring patterns to be chosen
- A constrained event of definite or variable duration take place at some unspecified time
- One party to propose an event to another party and require that they respond within a particular time window if they agree.
- Require another party to "confirm" their previous acceptance by some time, otherwise the original acceptance is not valid.

CalWS Requests/Responses

Calling Pattern

The general calling pattern for CalWS is

- A client generates a CalWS request with a single CalWS method in it.
- The CalWS service generates a response.

CalWS Request

Request Body

The body of the request contains the particular method (see Methods below) being called along with necessary data. In this example, the GetItem method is used:

```
<Body>
        <GetItems>
                <!—GetItems info>
        </GetItems>
```

```
</Body>
```

CalWS Response

The body of the response contains response data for each method called in the request. In this example, the GetItem method is used again. Note that each request method has "Response" appended to the method name for its corresponding response element:

```
<Body>
        <GetItemResponse>
                <!—GetItemResponse info>
        </GetItemResponse>
<Body>
```

Authentication and Authorization

Details surrounding authentication of CalWS requests and authorization of access to particular calendar resources is outside the scope of this document.

Basic Methods

CreateCalendar

Description: Creates a calendar with a unique id and optional display name and description.

CreateCalendar Request:

Example:

```
<CreateCalendar>
        <Calendar>
                <DisplayName>Utility Rate Schedule</DisplayName>
                <Description>Rate schedule for May 17, 2010</Description>
        </Calendar>
</CreateCalendar>
```

CreateCalendar Response:

Example:

```
<CreateCalendarResponse>
```

```
        <Calendar Id="12345">

        </Calendar>

</CreateCalendarResponse>
```

DeleteCalendar

Description: Deletes a calendar specified by a unique id.

UpdateCalendar

Description: Updates display name or description for a calendar specified by a unique id.

CreateItem

Description: Creates a calendar item on a specific calendar and using unique id specified for the calendar item.

CreateItem Request:

Example:

```
<CreateItem>

        <Calendar Id="12345"/>

        <Items>

                <!—Items represented as ICalendar in XML>

                <icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0”>

                <vcalendar>

                <properties>

                        <version><text>2.0</text></version>

                </properties>

                <components>

                        <vevent>

                        <properties>

                                <dtstart>20100517T080000Z</dtstart>

                                <dtend>20100517T170000Z</dtend>

                                <summary>

                                        <text>Rate info May 17-21, 2010</text>

                                </summary>
```

```
                        <uid>
                            <text>4088E990AD89CB3DBB484909</text>
                        </uid>
                    </properties>
                    </vevent>
                </components>
                </vcalendar>
                </icalendar>
            </Items>
</CreateItem>
```

CreateItem Response:

Example:

```
<CreateItemResponse>
</CreateItemResponse>
```

GetItems

Description: Gets properties for one or more calendar items using a unique id specified for each calendar item.

A unique id must be specified for each calendar item to be retrieved.

GetItems Request:

Example:

```
<GetItem>
        <ItemIds>
            <ItemId Id="56789"/>
            <ItemId Id="56790"/>
        </ItemIds>
</GetItem>
```

GetItems Response:

Example: TBD

```
<GetItemsResponse>

</GetItemsResponse>
```

GetItemsInRange

Description: Gets calendar items from a specific calendar that fall within a specific time range. Results will include items where part of the event lies outside the time range.

GetItemsInRange Request:

Example:

```
<GetItemsInRange>
        <Calendar Id="12345"/>
        <Range>
                <!—Range (May 17, 2010 8a-5p) represented as ICalendar in XML>
                <icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0”>

                <vcalendar>
                <properties>
                        <version><text>2.0</text></version>
                </properties>
                <components>
                        <vevent>
                        <properties>
                            <dtstart>20100517T080000Z</dtstart>
                            <dtend>20100517T170000Z</dtend>
                        </properties>
                        </vevent>
```

```
                </components>

                </vcalendar>

                </icalendar>

        </Range>

</GetItemsInRange>
```

GetItemsInRange Response:

Example:

```
<GetItemsInRangeResponse>

        <Calendar Id="12345"/>

        <Items>

                <!—Items in range represented as ICalendar in XML>

        </Items>

</GetItemsInRangeResponse>
```

DeleteItem

Description: Deletes calendar items (identified by unique ids) from a specific calendar.

UpdateItem

Description: Updates calendar items (identified by unique ids) on a specific calendar.

References

XCal – The XML Format for ICalendar

OASIS Working Draft - WS Calendar 1.0

EIS Alliance Customer Use Cases

Revisions

Changes from -00

- Changed datetime format in examples to match xCal.
- Added schema reference to xCal in examples.
- Updated section on requirements for extensions to xCal.

Changes from -01

- Introduced “CalWS” as working name to avoid confusion with OASIS document.
- Updated discussion on precision, uncertainty, sequence

Changes from -02

- Added "Other Protocols" section..