Appears within a link relation describing collections or entities. The property takes no value and indicates that this child element is a calendar collection.

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/calendar-collection"
          xsi:nil="true" />
```

8.21 CalWS:privilege-set XML element

http://docs.oasis-open.org/ns/wscal/calws:privilege-set

Appears within a link relation describing collections or entities and specifies the set of privileges allowed to the current authenticated principal for that collection or entity.

```
<!ELEMENT calws:privilege-set (calws:privilege*)>
<!ELEMENT calws:privilege ANY>
```

Each privilege element defines a privilege or access right. The following set is currently defined

- CalWS: Read - current principal has read access
- CalWS: Write - current principal has write access

```
<calWS:privilege-set>
  <calWS:privilege><calWS:read></calWS:privilege>
  <calWS:privilege><calWS:write></calWS:privilege>
</calWS:privilege-set>
```

9 Retrieving Collection and Service Properties

Properties, related services and locations are obtained from the service or from service resources in the form of an XRD document as defined by [XRD-1.0].

Given the URL of a CalWS service a client retrieves the service XRD document through a GET on the service URL with an ACCEPT header specifying application/xrd+xml.

Retrieving resource properties is identical to obtaining service properties, that is, execute a GET on the target URL with an ACCEPT header specifying application/xrd+xml.

The service properties define the global limits and defaults. Any properties defined on collections within the service hierarchy override those service defaults. The service may choose to prevent such overriding of defaults and limits when appropriate.

9.1 Request parameters

- None

9.2 Responses:

- 200: OK
- 403: Forbidden
- 404: Not found

9.3 Example - retrieving server properties:

```
>>Request

GET / HTTP/1.1
Host: example.com
ACCEPT:application/xrd+xml

>>Response
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Expires>1970-01-01T00:00:00Z</Expires>
    <Subject>http://example.com/calws</Subject>
    <Property type="http://docs.oasis-open.org/ns/wscal/calws/created"
         >1970-01-01</Property>

    <Link rel="http://docs.oasis-open.org/ns/wscal/calws/timezone-service"
        href="http://example.com/tz" />

    <calWS:privilege-set>
    <calWS:privilege><calWS:read></calWS:privilege>
    </calWS:privilege-set>

    <Link rel="http://docs.oasis-open.org/ns/wscal/calws/principal-home"
        type="collection"
        href="http://example.com/calws/user/fred">
    <Title xml:lang="en">Fred's calendar home</Title>
    </Link>

    <Link rel="http://docs.oasis-open.org/ns/wscal/calws/child-collection"
        type="calendar,scheduling"
        href="http://example.com/calws/user/fred/calendar">
    <Title xml:lang="en">Calendar</Title>
```

```
    </Link>

    <Property type="http://docs.oasis-open.org/ns/wscal/calws/max-instances"
        >1000</Property>

    <Property type="http://docs.oasis-open.org/ns/wscal/calws/max-attendees-
per-instance"
        >100</Property>

</XRD>
```

10 Creating Calendar Object Resources

Creating calendar object resources is carried out by a POST on the parent collection. The body of the request will contain the resource being created. The request parameter "action=create" indicates this POST is a create. The location header of the response gives the URL of the newly created object.

10.1 Request parameters

- action=create

10.2 Responses:

- 201: created
- 403: Forbidden - no access

10.3 Preconditions for Calendar Object Creation

- **CalWS:target-exists**: The target of a PUT must exist. Use POST to create entities and PUT to update them.
- **CalWS:not-calendar-data:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST be a supported media type (i.e., iCalendar) for calendar object resources;
- **CalWS:invalid-calendar-data:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST be valid data for the media type being specified (i.e., MUST contain valid iCalendar data);
- **CalWS:invalid-calendar-object-resource:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST obey all restrictions specified in Calendar Object Resources (e.g., calendar object resources MUST NOT contain more than one type of calendar component, calendar object resources MUST NOT specify the iCalendar METHOD property, etc.);
- **CalWS:unsupported-calendar-component:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST contain a type of calendar component that is supported in the targeted calendar collection;
- **CalWS:uid-conflict:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST NOT specify an iCalendar UID property value already in use in the targeted calendar collection or overwrite an existing calendar object resource with one that has a different UID property value. Servers SHOULD report the URL of the resource that is already making use of the same UID property value in the CalWS:href element
  <!ELEMENT uid-conflict (CalWS:href)>
- **CalWS:invalid-calendar-collection-location:** In a COPY or MOVE request, when the Request-URI is a calendar collection, the Destination-URI MUST identify a location where a calendar collection can be created;
- **CalWS:exceeds-max-resource-size:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have an octet size less than or equal to the value of the CalDAV:max-resource-size property value on the calendar collection where the resource will be stored;
- **CalWS:before-min-date-time:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have all of its iCalendar DATE or DATE-TIME property values (for each recurring instance) greater than or equal to the value of the CalDAV:min- date-time property value on the calendar collection where the resource will be stored;

- **CalWS:after-max-date-time:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have all of its iCalendar DATE or DATE-TIME property values (for each recurring instance) less than the value of the CalDAV:max-date-time property value on the calendar collection where the resource will be stored;
- **CalWS:too-many-instances:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST generate a number of recurring instances less than or equal to the value of the CalDAV: max-instances property value on the calendar collection where the resource will be stored;
- **CalWS:too-many-attendees-per-instance:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have a number of ATTENDEE properties on any one instance less than or equal to the value of the CalDAV:max-attendees-per-instance property value on the calendar collection where the resource will be stored;

10.4 Example - successful POST:

```
>>Request

POST /user/fred/calendar/?action=create HTTP/1.1
Host: example.com
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
  ...
  </vcalendar>
</icalendar>

>>Response

HTTP/1.1 201 Created
Location: http://example.com/user/fred/calendar/event1.ics
```

10.5 Example - unsuccessful POST:

```
>>Request

POST /user/fred/readcalendar/?action=create HTTP/1.1
Host: example.com
Content-Type: text/text; charset="utf-8"
Content-Length: ?

This is not an xml calendar object

>>Response

HTTP/1.1 403 Forbidden
 <?xml version="1.0" encoding="utf-8"
      xmlns:D="DAV:"
      xmlns:C="urn:ietf:params:xml:ns:caldav" ?>
 <D:error>
     <C:supported-calendar-data/>
     <D:description>Not an icalendar object</C:description>
 </D:error>
```

11 Retrieving resources

A simple GET on the href will return a named resource. If that resource is a recurring event or task with overrides, the entire set will be returned. The desired format is specified in the ACCEPT header. The default form is application/xml+calendar

11.1 Request parameters

- none

11.2 Responses:

- 200: OK
- 403: Forbidden - no access
- 406 The requested format specified in the accept header is not supported.

11.3 Example - successful fetch:

```
>>Request

GET /user/fred/calendar/event1.ics HTTP/1.1
Host: example.com

>>Response

HTTP/1.1 200 OK
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
    ...
  </vcalendar>
</icalendar>
```

11.4 Example - unsuccessful fetch:

```
>>Request

PUT /user/fred/calendar/noevent1.ics HTTP/1.1
Host: example.com

>>Response

HTTP/1.1 404 Not found
```

12 Updating resources

Resources are updated with the PUT method targeted at the resource href. The body of the request contains a complete new resource which effectively replaces the targeted resource. To allow for optimistic locking of the resource use the if-match header.

When updating a recurring event all overrides and master must be supplied as part of the content.

Preconditions as specified in Section 10.3 are applicable.

12.1 Responses:

- 200: OK
- 304: Not modified - entity was modified by some other request
- 403: Forbidden - no access, does not exist etc. See error response

Example 16: Successful Update

```
>>Request

PUT /user/fred/calendar/event1.ics HTTP/1.1
Host: example.com
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
     ...
  </vcalendar>
</icalendar>

>>Response

HTTP/1.1 200 OK
```

Example 17: Unsuccessful Update

```
>>Request

PUT /user/fred/readcalendar/event1.ics HTTP/1.1
Host: example.com
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
    ...
  </vcalendar>
</icalendar>

>>Response

HTTP/1.1 403 Forbidden
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8"
```

```
      xmlns:D="DAV:"
      xmlns:CW=" http://docs.oasis-open.org/ws-calendar/CalWS" ?>
<CW:error>
  <CW:target-exists/>
  <CW:description>Target of update must exist</C:description>
</CW:error>
```

13 Deletion of resources

Delete is defined in **[RFC 2616]** Section 9.7. In addition to conditions defined in that specification, servers must remove any references from the deleted resource to other resources. Resources are deleted with the DELETE method targeted at the resource URL. After a successful completion of a deletion a GET on that URL must result in a 404 - Not Found status.

13.1 Delete for Collections

Delete for collections may or may not be supported by the server. Certain collections are considered undeletable. On a successful deletion of a collection all contained resources to any depth must also be deleted.

13.2 Responses:

- 200: OK
- 403: Forbidden - no access
- 404: Not Found

14 Querying calendar resources

Querying provides a mechanism by which information can be obtained from the service through possibly complex queries. A list of icalendar properties can be specified to limit the amount of information returned to the client. A query takes the parts

- Limitations on the data returned
- Selection of the data
- Optional timezone id for floating time calculations.

The current specification uses CalDAV multiget and calendar-query XML bodies as specified in **[RFC 4791]** with certain limitations and differences.

1. The POST method is used for all requests, the action being identified by the outer element.
2. While CalDAV servers generally only support [RFC 5545] and assume that as the default, the delivery format for CalWS will, by default, be [draft-xcal].
3. The CalDAV query allows the specification of a number of DAV properties. Specification of these properties, with the exception of DAV:getetag, is considered an error in CalWS.
4. The CalDAV:propnames element is invalid

With those differences, the CalDAV specification is the normative reference for this operation.

14.1 Limiting data returned

This is achieved by specifying one of the following

- CalDAV:allprop return all properties (some properties are specified as not being part of the allprop set so are not returned)
- CalDAV:prop An element which contains a list of properties to be returned . May only contain DAV:getetag and CalDAV:calendar-data

Of particular interest, and complexity, is the calendar-data property which can contain a time range to limit the range of recurrences returned and/or a list of calendar properties to return.

14.2 Pre/postconditions for calendar queries

The preconditions as defined in in **[RFC 4791]** Section 7.8 apply here. CalDav errors may be reported by the service when preconditions or postconditions are violated.

14.3 Example: time range limited retrieval

This example shows the time-range limited retrieval from a calendar which results in 2 events, one a recurring event and one a simple non-recurring event.

```
>> Request <<

POST /user/fred/calendar/ HTTP/1.1
Host: calws.example.com
Depth: 1
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:calendar-query xmlns:D="DAV:"
                  xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:prop>
    <D:getetag/>
    <C:calendar-data content-type="application/xml+calendar" >
```

```
      <C:comp name="VCALENDAR">
        <C:prop name="VERSION"/>
        <C:comp name="VEVENT">
          <C:prop name="SUMMARY"/>
          <C:prop name="UID"/>
          <C:prop name="DTSTART"/>
          <C:prop name="DTEND"/>
          <C:prop name="DURATION"/>
          <C:prop name="RRULE"/>
          <C:prop name="RDATE"/>
          <C:prop name="EXRULE"/>
          <C:prop name="EXDATE"/>
          <C:prop name="RECURRENCE-ID"/>
        </C:comp>
      </C:comp>
    </C:calendar-data>
  </D:prop>
  <C:filter>
    <C:comp-filter name="VCALENDAR">
      <C:comp-filter name="VEVENT">
        <C:time-range start="20060104T000000Z"
                      end="20060105T000000Z"/>
      </C:comp-filter>
    </C:comp-filter>
  </C:filter>
</C:calendar-query>

>> Response <<

HTTP/1.1 207 Multi-Status
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
               xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:response>
    <D:href>http://cal.example.com/bernard/work/abcd2.ics</D:href>
    <D:propstat>
      <D:prop>
        <D:getetag>"fffff-abcd2"</D:getetag>
        <C:calendar-data content-type="application/xml+calendar" >
          <xc:icalendar
             xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
   <xc:vcalendar>
     <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
       <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
     </xc:properties>
     <xc:components>
      <xc:vevent>
       <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060102T120000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
```

```
          <xc:text>Event #2</xc:text>
        </xc:summary>
        <xc:uid>
          <xc:text>00959BC664CA650E933C892C@example.com</xc:text>
        </xc:uid>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
      </xc:properties>
    </xc:vevent>

    <xc:vevent>
      <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060104T140000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
          <xc:text>Event #2 bis</xc:text>
        </xc:summary>
        <xc:uid>
          <xc:text>00959BC664CA650E933C892C@example.com</xc:text>
        </xc:uid>
        <xc:recurrence-id>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060104T120000</xc:date-time>
        </xc:recurrence-id>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
      </xc:properties>
    </xc:vevent>

    <xc:vevent>
      <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060106T140000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
          <xc:text>Event #2 bis bis</xc:text>
        </xc:summary>
        <xc:uid>
          <xc:text>00959BC664CA650E933C892C@example.com</xc:text>
        </xc:uid>
        <xc:recurrence-id>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060106T120000</xc:date-time>
```

```
        </xc:recurrence-id>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
       </xc:properties>
      </xc:vevent>
     </xc:components>
    </xc:vcalendar>
   </xc:icalendar>
          </C:calendar-data>
        </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
    </D:response>
    <D:response>
      <D:href>http://cal.example.com/bernard/work/abcd3.ics</D:href>
      <D:propstat>
        <D:prop>
          <D:getetag>"fffff-abcd3"</D:getetag>
          <C:calendar-data content-type="application/xml+calendar" >
            <xcal:icalendar
               xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
    <xc:vcalendar>
     <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
       <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
     </xc:properties>
     <xc:components>
      <xc:vevent>
       <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060104T100000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
         <xc:text>Event #3</xc:text>
        </xc:summary>
        <xc:uid>
         <xc:text>DC6C50A017428C5216A2F1CD@example.com</xc:text>
        </xc:uid>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
       </xc:properties>
      </xc:vevent>
     </xc:components>
    </xc:vcalendar>
   </xc:icalendar>
        </C:calendar-data>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
```

```
  </D:response>
</D:multistatus>
```

15 Free-busy queries

Free-busy queries are used to obtain free-busy information for a calendar-collection or principals. The result contains information only for events to which the current principal has sufficient access.

When targeted at a calendar collection the result is based only on the calendaring entities contained in that collection. When targeted at a principal free-busy URL the result will be based on all information which affect the principals free-busy status, for example availability.

The possible targets are:

- A calendar collection URL
- The XRD link with relation CalWS/current-principal-freebusy
- The XRD link with relation CalWS/principal-freebusy with a principal given in the request.

The query follows the specification defined in Error! Reference source not found. with certain limitations. As an authenticated user to the CalWS service scheduling read-freebusy privileges must have been granted. As an unauthenticated user equivalent access must have been granted to unauthenticated access.

Freebusy information is returned by default as xcalendar vfreebusy components, as defined by **[draft-xcal]**. Such a component is not meant to conform to the requirements of VFREEBUSY components in **[RFC 5546]**. The VFREEBUSY component SHOULD conform to section "4.6.4 Free/Busy Component" of **[RFC 5545]**. A client SHOULD ignore the ORGANIZER field..

Since a Freebusy query can only refer to a single user, a client will already know how to match the result component to a user. A server MUST only return a single vfreebusy component.

15.1 ACCEPT header

The Accept header is used to specify the format for the returned data. In the absence of a header the data should be returned as specified in **[draft-xcal]**, that is, as if the following had been specified

```
ACCEPT: application/xml+calendar
```

15.2 URL Query Parameters

None of these parameters are required except for the conditions noted below. Appropriate defaults will be supplied by the server.

15.2.1 start

Default: The default value is left up to the server. It may be the current day, start of the current month, etc.

Description: Specifies the start date for the Freebusy data. The server is free to ignore this value and return data in any time range. The client must check the data for the returned time range.

Format:A profile of an **[RFC3339]** Date/Time. Fractional time is not supported. The server MUST support the expanded version e.g.

```
2007-01-02T13:00:00-08:00
```

It is up to the server to interpret local date/times.

Example:

```
2007-02-03T15:30:00-0800
2007-12-01T10:15:00Z
```

Notes: Specifying only a start date/time without specifying an end-date/time or period should be interpreted as in **[RFC 5545**]. The effective period should cover the remainder of that day.

Date-only values are disallowed as the server cannot determine the correct start of the day. Only

UTC or date/time with offset values are permitted.

15.2.2 end

Default: Same as start

Description: Specifies the end date for the Freebusy data. The server is free to ignore this value.

Format: Same as start

Example: Same as start

15.2.3 period

Default: The default value is left up to the server. The recommended value is "P42D".

Description: Specifies the amount of Freebusy data to return. A client cannot specify both a period and an end date. Period is relative to the start parameter.

Format: A duration as defined in section 4.3.6 of **[RFC 5545]**

Example:

```
P42D
```

15.2.4 account

Default: none

Description: Specifies the principal when the request is targeted at the XRD CalWS/principal-freebusy. Specification of this parameter is an error otherwise.

Format: Server specific

Example:

```
fred
/principals/users/jim
user1@example.com
```

15.3 URL parameters - notes

The server is free to ignore the start, end and period parameters. It is recommended that the server return at least 6 weeks of data from the current day.

A client MUST check the time range in the VFREEBUSY response as a server may return a different time range than the requested range.

15.4 HTTP Operations

The server SHOULD return an Etag response header for a successful GET request targeting a Freebusy read URL. Clients MAY use the Etag response header value to do subsequent "conditional" GET requests that will avoid re-sending the Freebusy data again if it has not changed.

15.5 Response Codes

Below are the typical status codes returned by a GET request targeting a Free-busy URL. Note that other HTTP status codes not listed here might also be returned by a server.

- 200 OK
- 302 Found
- 400 Start parameter could not be understood / End parameter could not be understood / Period parameter could not be understood
- 401 Unauthorized

- 403 Forbidden
- 404 The data for the requested principal is not currently available, but may be available later.
- 406 The requested format in the accept header is not supported.
- 410 The data for the requested principal is no longer available
- 500 General server error

15.6 Examples

The following are examples of URLs used to retrieve Free-busy data for a user:

```
http://www.example.com/freebusy/user1@example.com?
start=2007-09-01T00:00:00-08:00&end=2007-09-31T00:00:00-08:00

http://www.example.com/freebusy/user1@example.com?
start=2007-09-01T00:00:00-08:00&end=2007-09-31T00:00:00-08:00

http://www.example.com/freebusy/user1@example.com

http://www.example.com/freebusy?user=user%201@example.com&
start=2008-01-01T00:00:00Z&end=2008-12-31T00:00:00Z
```

Some Request/Response Examples:

A URL with no query parameters:

```
>> Request <<
GET /freebusy/bernard/ HTTP/1.1
Host: www.example.com

>> Response <<
HTTP/1.1 200 OK
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: xxxx

<xc:icalendar xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
  <xc:vcalendar>
    <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
        <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
    </xc:properties>
    <xc:components>
      <xc:vfreebusy>
        <xc:properties>
          <xc:uid>
            <xc:text>76ef34-54a3d2@example.com</xc:text>
          </xc:uid>
          <xc:dtstart>
            <xc:date-time>20060101T000000Z</xc:date-time>
          </xc:dtstart>
          <xc:dtend>
            <xc:date-time>20060108T000000Z</xc:date-time>
          </xc:dtend>
          <xc:dtstamp>
            <xc:date-time>20050530T123421Z</xc:date-time>
          </xc:dtstamp>
          <xc:freebusy>
            <xc:parameters>
              <xc:fbtype>BUSYTENTATIVE<xc:fbtype>
            <xc:parameters>
            <xc:period>20060102T100000Z/20060102T120000Z</xc:period>
```

```
          </xc:freebusy>
          <xc:freebusy>
            <xc:period>20060103T100000Z/20060103T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:period>20060104T100000Z/20060104T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:parameters>
              <xc:fbtype>BUSYUNAVAILABLE<xc:fbtype>
            <xc:parameters>
            <xc:period>20060105T100000Z/20060105T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:period>20060106T100000Z/20060106T120000Z</xc:period>
          </xc:freebusy>
      </xc:vfreebusy>
    </xc:components>
  </xc:vcalendar>
<xc:icalendar>
```

A URL with start and end parameters:

```
>> Request <<
GET /freebusy/user1@example.com?start=2007-09-01T00:00:00-08:00&end=2007-09-
31T00:00:00-08:00
HTTP/1.1
Host: www.example.com

>> Response <<
HTTP/1.1 200 OK
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: xxxx

<xc:icalendar xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
  <xc:vcalendar>
    <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
        <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
    </xc:properties>
    <xc:components>
      <xc:vfreebusy>
        <xc:properties>
          <xc:uid>
            <xc:text>76ef34-54a3d2@example.com</xc:text>
          </xc:uid>
          <xc:dtstart>
            <xc:date-time>20070901T000000Z</xc:date-time>
          </xc:dtstart>
          <xc:dtend>
            <xc:date-time>20070931T000000Z</xc:date-time>
          </xc:dtend>
          <xc:dtstamp>
            <xc:date-time>20050530T123421Z</xc:date-time>
          </xc:dtstamp>
          <xc:freebusy>
            <xc:period>20070915T230000Z/20070916T010000Z</xc:period>
          </xc:freebusy>
      </xc:vfreebusy>
    </xc:components>
  </xc:vcalendar>
<xc:icalendar>
```

A URL for which the server does not have any data for that user:

```
>> Request <<
GET /freebusy/user1@example.com?start=2012-12-01T00:00:00-08:00&end=2012-12-
31T00:00:00-08:00
HTTP/1.1
Host: www.example.com

>> Response <<
HTTP/1.1 404 No data
```

16 Conformance

WS-Calendar Intervals SHALL have a Duration. Intervals MAY have a StartTime. Intervals SHALL NOT include an END time. If a non-compliant Interval is received with an END time, it may be ignored.

A performance component SHALL not include Start, Stop, and Duration elements. Two out of the three elements is acceptable, but not three.

In Partitions, the Description, Summary and Priority of each Interval SHALL be excluded.

An Calendar Gluon may either have a dtStart or a dtEnd, but may not have both.

All OASIS specifications require conformance

A. Acknowledgements

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

Participants:

Bruce Bartell, Southern California Edison
Brad Benson, Trane
Edward Cazalet, Individual
Toby Considine, University of North Carolina at Chapel Hill
William Cox, Individual
Sharon Dinges, Trane
Craig Gemmill, Tridium, Inc.
Girish Ghatikar, Lawrence Berkeley National Laboratory
Gerald Gray, Southern California Edison
Gale Horst, Electric Power Research Institute (EPRI)
Gershon Janssen, Individual
Ed Koch, Akuacom Inc.
Benoit Lepeuple, LonMark International*
Carl Mattocks, CheckMi*
Robert Old, Siemens AG
Alexander Papaspyrou, Technische Universitat Dortmund
Jeremy J. Roberts, LonMark International
David Thewlis, CalConnect

The Calendaring and Scheduling Consortium (CalConnect) TC-XML committee worked closely with WS-Calendar Technical Committee, bridging to developing IETF standards and contributing the Services definitions that make up Services beginning in Section 7, Calendar Services. The Technical Committee gratefully acknowledges their assistance and cooperation as well. Contributors to TC XML include:

Cyrus Daboo, Apple
Mike Douglas, Rensselaer Polytechnic Institute
Steven Lees, Microsoft
Tong Li, IBM

B. An Introduction to Internet Calendaring

The WS-Calendar Technical Committee thanks CalConnect for contributing this overview of iCalendar and its use.

B.1 icalendar

B.1.1 History

The iCalendar specification was first produced by the IETF in 1998 as RFC 2445 [1]. Since then it has become the dominant standard for calendar data interchange on the internet and between devices (desktop computers, mobile phones etc.). The specification was revised in 2009 as RFC 5545 [4].

Alongside iCalendar is the iTIP specification (RFC 2446 [2] and revised as RFC 5546[5]) that defines how iCalendar is used to carry out scheduling operations (for example, how an organizer can invite attendees to a meeting and receive their replies). This forms the basis for email-based scheduling using iMIP (the specification that describes how to use iTIP with email - RFC 2447 [3]).

iCalendar itself is a text-based data format. However, an XML format is also available, providing a one-to-one mapping to the text format (draft [7]).

iCalendar data files typically have a .ics file name extension. Most desktop calendar clients can import or export iCalendar data, or directly access such data over the Internet using a variety of protocols.

B.1.2 Data model

The iCalendar data format has a well defined data model. "iCalendar objects" encompass a set of "iCalendar components" each of which contains a set of "iCalendar properties" and possibly other sub-components. An iCalendar property consists of a name, a set of optional parameters (specified as "key-value" pairs) and a value.

iCalendar components include:

"VEVENT" which represents an event

"VTODO" which represents a task or to-do

"VJOURNAL" which represents a journal entry

"VFREEBUSY" which represents periods of free or busy time information

"VTIMEZONE" which represents a timezone definition (timezone offset and daylight saving rules)

"VALARM" is currently the only defined sub-component and is used to set alarms or reminders on events or tasks.

Properties include:

"DTSTART" which represents a start time for a component

"DTEND" which represents an end time for a component

"SUMMARY" which represents a title or summary for a component

"RRULE" which can specify rules for repeating events or tasks (for example, every day, every week on Tuesdays, etc.)

"ORGANIZER" which represents the calendar user who is organizing an event or assigning a task

"ATTENDEE" which represents calendar users attending an event or assigned a task

In addition to this data model and the pre-defined properties, the specification defines how all those are used together to define the semantics of calendar objects and scheduling. The semantics are basically a set of rules stating how all the components and properties are used together to ensure that all iCalendar products can work together to achieve good interoperability. For example, a rule requires that all events must have one and only one "DTSTART" property. The most important part of the iCalendar specification

is the semantics of the calendaring model that it represents. The use of text or XML to encode those is secondary.

B.1.3 Scheduling

The iTIP specification defines how iCalendar objects are exchanged in order to accomplish the key task needed to schedule events or tasks. An example of a simple workflow is as follows:

1. To schedule an event, an organizer creates the iCalendar object representing the event and adds calendar users as attendees.
2. The organizer then sends an iTIP "REQUEST" message to all the attendees.
3. Upon receipt of the scheduling message, each attendee can decide whether they want to attend the meeting or not.
4. Each attendee can then respond back to the organizer using an iTIP "REPLY" message indicating their own attendance status.

iTIP supports other types of scheduling messages, for example, to cancel meetings, add new instances to a repeating meeting, etc.

B.1.4 Extensibility

iCalendar was designed to be extensible, allowing for new components, properties and parameters to be defined as needed. A registry exists to maintain the list of standard extensions with references to their definitions to ensure anyone can use them and work well with others.

B.2 Calendar data access and exchange protocols

B.2.1 Internet Calendar Subscriptions

An Internet calendar subscription is simply an iCalendar data file made available on a web server. Users can use this data in two ways:

- The data can be downloaded from the web server and then imported directly into an iCalendar aware client. This solution works well for calendar data that is not likely to change over time (for example the list of national holidays for the next year).
- Calendar clients that support "direct" subscriptions can use the URL to the calendar data on the web server to download the calendar data themselves. Additionally, the clients can check the web server on a regular basis for updates to the calendar data, and then update their own cached copy of it. This allows calendar data that changes over time to be kept synchronized.

B.2.2 CalDAV

CalDAV is a calendar access protocol and is defined in RFC 4791 [6]. The protocol is based on WebDAV which is an extension to HTTP that provides enhanced capabilities for document management on web servers.

CalDAV is used in a variety of different environments, ranging from very large internet service providers, to large and small corporations or institutions, and to small businesses and individuals.

CalDAV clients include desktop applications, mobile devices and browser-based solutions. It can also be used by "applets", for example, a web page panel that displays a user's upcoming events.

One of the key aspects of CalDAV is its data model. Simply put, it defines a "calendar home" for each calendar user, within which any number of "calendars" can be created. Each "calendar" can contain any number of iCalendar objects representing individual events, tasks or journal entries. This data model ensures that clients and servers can interoperate well.

In addition to providing simple operations to read, write and delete calendar data, CalDAV provides a querying mechanism to allow clients to fetch calendar data matching specific criteria. This is commonly

used by clients to do "time-range" queries, i.e., find the set of events that occur within a given start/end time period.

CalDAV also supports access control allowing for features such as delegated calendars and calendar sharing.

CalDAV also specifies how scheduling operations can be done using the protocol. Whilst it uses the semantics of the iTIP protocol, it simplifies the process by allowing simple calendar data write operations to trigger the sending of scheduling messages, and it has the server automatically process the receipt of scheduling messages. Scheduling can be done with other users on the CalDAV server or with calendar users on other systems (via some form of "gateway").

B.2.3 ActiveSync/SyncML

ActiveSync and SyncML are technologies that allow multiple devices to synchronize data with a server, with calendar data being one of the classes of data supported. These have typically been used for low-end and high-end mobile devices.

B.2.4 CalWS

CalWS is a web services calendar access API developed by The Calendaring and Scheduling Consortium and the OASIS organization, to be used as part of the Oasis WS-Calendar standard. It provides an API to access and manipulate calendar data stored on a server. It follows a similar data model to CalDAV and has been designed to co-exist with a CalDAV service offering the same data.

B.2.5 iSchedule

iSchedule is a protocol to allow scheduling between users on different calendaring systems and across different internet domains. It transports iTIP scheduling messages using HTTP between servers. Servers use DNS and various security mechanisms to determine the authenticity of messages received.

It has been specifically designed to be independent of any calendar system in use at the endpoints, so that it is compatible with many different systems. This allows organizations with different calendar systems to exchange scheduling messages with each other, and also allows a single organization with multiple calendar systems (for example due to mergers, or different departmental requirements) to exchange scheduling messages between users of each system.

B.3 References

[1] https://datatracker.ietf.org/doc/rfc2445/ : ‘Internet Calendaring and Scheduling Core Object Specification’

[2] https://datatracker.ietf.org/doc/rfc2446/ :‘iCalendar Transport-Independent Interoperability Protocol’

[3] https://datatracker.ietf.org/doc/rfc2447/ : ‘iCalendar Message-Based Interoperability Protocol’

[4] https://datatracker.ietf.org/doc/rfc5545/ :‘Internet Calendaring and Scheduling Core Object Specification’

[5] https://datatracker.ietf.org/doc/rfc5546/ : ‘iCalendar Transport-Independent Interoperability Protocol’

[6] https://datatracker.ietf.org/doc/rfc4791/ : ‘Calendaring Extensions to WebDAV’

[7] https://datatracker.ietf.org/doc/draft-daboo-et-al-icalendar-in-xml/ : ‘xCal: The XML format for iCalendar’

C. Overview of WS-Calendar, its Antecedents and its Use

iCalendar has long been the predominant message format for an Internet user to send meeting requests and tasks to other Internet users by email. The recipient can respond to the sender easily or counter propose another meeting date/time. iCalendar support is built into all major email systems and email clients. While SMTP is the predominant means to transport iCalendar messages, protocols including WebDAV and SyncML are used to transport collections of iCalendar information. No similar standard for service interactions has achieved similar widespread use.

The Calendar and Scheduling Consortium (CalConnect), working within the IETF, updated the iCalendar standard in the summer of 2009 to support extension ([RFC5545]). In 2010, the same group defined [XCAL], a canonical XML serialization for iCalendar, currently (08/21/2008) on the recommended standards track within the IETF. This specification supports extensions, including handling non-standard, i.e., non-iCalendar, data during message storage and retrieval.

WS-Calendar builds on this work, and consists of extensions to the vocabulary of iCalendar, along with standard services to extend calendaring and scheduling into service interactions. iCalendar consists of a number of fields that support the delivery, update, and synchronization of if calendar messages and a list of components. The components can specify defined relationships between each other.

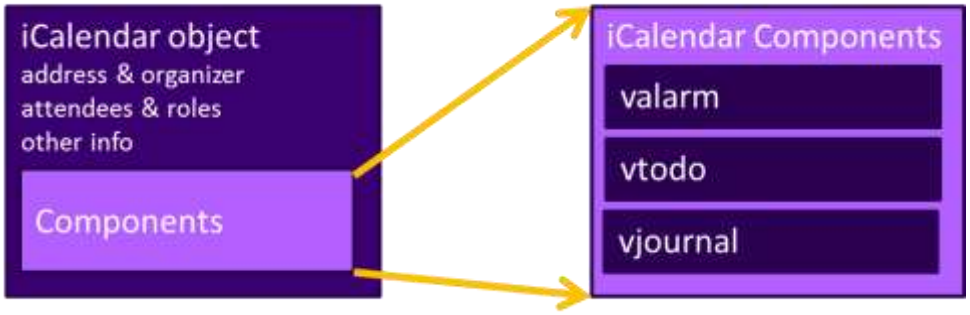

Figure 3: iCalendar overview

WS-Calendar defines the Interval, a profile of the vtodo component requiring only a duration and an artifact to define service delivery and performance. WS-Calendar also defines the CalendarGluon component, a container for holding only a service delivery and performance artifact, to associate with a component or group of components.

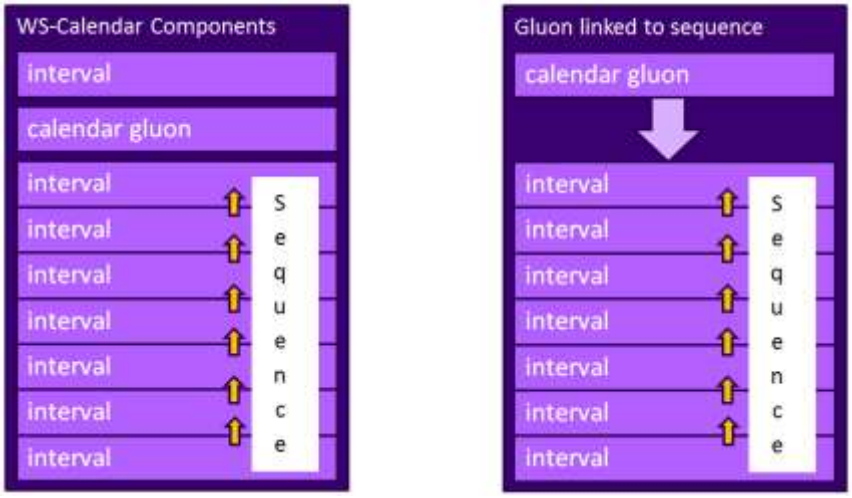

Figure 4: WS-Calendar and EMIX

A set of intervals that have defined temporal relationships is a Sequence. Temporal relationships express how the occurrence of one interval is related to another. For example, Interval B may begin 10 minutes after Interval A completes, or Interval D may start 5 minutes after Interval C starts. An Calendar Gluon linked to a Sequence defines service performance for all Intervals in the Sequence. Because each interval has its own service performance contract, specifications built on WS-Calendar can define rules for inheritance and over-rides with a sequence.

The Partition is a sub-class of a Sequence in which all Intervals follow consecutively with no lag time. Intervals in a Partition normally have the same Duration, but WS-Calendar does support overriding the duration on an individual basis.

C.1 Scheduling Sequences

A Sequence is a general pattern of behaviors and results that does not require a specific schedule. A publishing service may advertise a Sequence with no schedule, i.e., no specific time for performance. When the Sequence is invoked or contracted, a specific performance time is added. In the original iCalendar components, this would add the starting date and time (dtStart) to the component. In WS-Calendar, we add the starting date and time only to the first Interval of a Sequence; the performance times for all other Intervals in the Sequence are derived from that one start time.

C.1.1 Academic Scheduling example

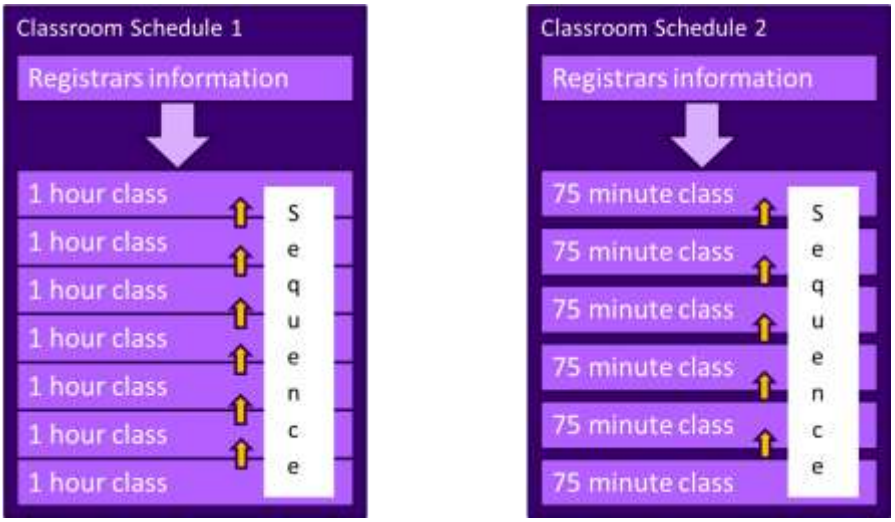

Figure 5: Classroom Scheduling Example

A college campus uses two schedules to schedule its buildings. In Schedule 1, classes start on the hour, and follow one after another; each class starts on the hour. In the second schedule, each class lasts an hour and a quarter, and there is a fifteen minute gap between classes; classes start on the half hour. On many campuses, the sequence in Schedule 1 may describe classes taught on Monday, Wednesday, and Friday. Schedule 2 may describe classes taught on Tuesday and Thursday.

The registrar's office knows some key facts about each classroom, including whether it hosts a class during a particular period, and the number of students that will be in that class. The college wishes to optimize the provision of building services for each class. Such services may include adequate ventilation and comfortable temperatures to assure alert students. Other services may ensure that the classroom projection systems and A/V support services are warmed up in advance of a class, or powered off when a classroom is vacant.

Although most classes meet over typical schedule for the week (M-W-F or Tu-Th), some classes may not meet on Friday, or may have a tutorial section one day a week. The registrar's system, ever mindful of student privacy, shares only minimal information with the building systems such as how many students will be supported.

The Registrar's system schedule building systems using the Calendar Gluon (registrar's information) and the student counts for each interval, and schedules the Sequence in classroom schedule 1 three days a week for the next 10 weeks. The Registrar's system also schedules the sequence in classroom schedule 2 two days a week, also for 10 weeks.

This example demonstrates a system (A) that offers services using either of two sequences. Another business system (B) with minimal knowledge of how (A) works determines the performance requirements for (A). The business system (B) communicates these expectations are by scheduling the Sequences offered by (A).

C.1.2 Market Performance schedule

A factory relies on an energy-intensive process which is performs twice a year for eight weeks. The factory has some flexibility about scheduling the process; it can perform the work in either the early morning or the early evening; it avoids the afternoon when energy costs are highest. The factory works up a detailed profile of when it will need energy to support this process.

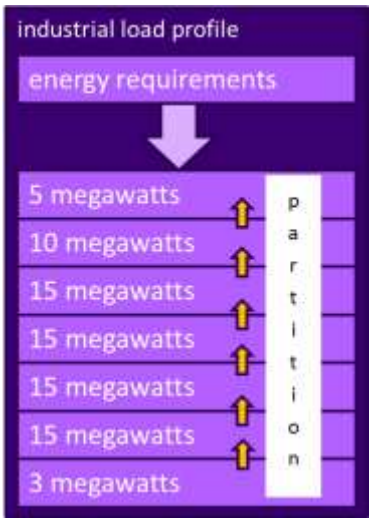

Figure 6: Daily Load Profile for Market Operations Example

Factory management has decided that they want to use only renewable energy products for this process. They approach two regional wind farms with the intent of making committed purchases of wind energy. The wind farms consider their proposals taking into account the seasonal weather forecasts they use to project their weather capacity, and considering the costs that may be required to buy additional wind energy on the spot market to make up any shortfalls.

Each energy supplier submits of the same sequence, a schedule, i.e. a daily starting time, and a price for the season's production. After considering the bids, and other internal costs of each proposal, the factory opts to accept a contract for the purchase of a fixed load profile (Partition), using the evening wind generation from one of the suppliers. This contract specifies Schedules of load purchases (starting data and time for the sequence) for each day.

2618 # D. Revision History

Revision	Date	Editor	Changes Made
1.0 WD 01	2010-03-11	Toby Considine	Initial document, largely derived from Charter
1.0 WD 02	2010-03-30	Toby Considine	Straw-man assertion of elements, components to push conversation
1.0 WD 03	2010-04-27	Toby Considine	Cleaned up Elements, added [XPOINTER] use, xs:duration elements
1.0 WD 04	2010-05-09	Toby Considine	Aligned Chapter 4 with the vAlarm and vToDo objects.
1.0 WD 05	2010-05-18	Toby Considine	Responded to comments, added references, made references to [XCAL] more consistent,
1.0 WD 06	2010-05-10	Toby Considine	Responded to comments from CalConnect, mostly constancy of explanations
1.0 WD 07	2010-07-28	Toby Considine	Incorporated input from informal public review, esp. SGIP PAP04. Firmed up relationships between scheduled objects
1.0 WD 08	2010-08-07	Toby Considine	Aligned with Interval / Partition / Sequence language. Reduced performance characteristics to before / after durations.
1.0 WD 09	2010-08-15	Toby Considine	Formalized Attachment section and rolled Performance into the Attachment. Created RelatedComponent object. Added CalWS Outline to specification. Removed SOOP section
1.0 WD 10	2010-08-28	Toby Considine, Benoit Lepeuple	Updated Time Stamp section Added background Appendices Incorporated Association language to replace RelatedComponent Recast examples to show inheritance, remove inconsistencies
1.0 WD 11	2010-09-11	Toby Considine	Traceability Release in support of a re-shuffling of the document. Sections 3, 4 were re-shuffled to create: 3: Interval / Relationships / Time Stamps 4: Performance / Attachments 5: Associations & Inheritance Also, changed all associations to Gluons. No paragraphs have been changed, just shuffled, changes accepted, to create clean base for editing
1.0 WD 12	2010-09-14	Toby Considine Dave Thewlis	Edits for clarity and flow following changes in WD11, updated examples based upon XSD artifacts. Adding final contribution from CalConnect for Services.

2619