Calendering Extensions
Internet-Draft
Intended status: Informational
Expires: January 24, 2018

C. Daboo
Apple
A. Quillaud
Oracle
K. Murchison, Ed.
FastMail
July 23, 2017

CalDAV Managed Attachments
draft-ietf-calext-caldav-attachments-03

Abstract

This specification defines an extension to the calendar access protocol (CalDAV) to allow attachments associated with iCalendar data to be stored and managed on the server.

This specification documents existing code deployed by multiple vendors. It is published as an Informational specification rather than Standards-Track due to its non-standard method of updating an existing resource via HTTP.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 24, 2018.

Table of Contents

1. Introduction

The iCalendar [RFC5545] data format is used to represent calendar data and is used with iTIP [RFC5546] to handle scheduling operations between calendar users.

[RFC4791] defines the CalDAV Calendar Access protocol, based on HTTP [RFC7230], for accessing calendar data stored on a server.

Calendar users often want to include attachments with their calendar data events or tasks (for example a copy of a presentation, or the meeting agenda). iCalendar provides an "ATTACH" property whose value is either the inline Base64 encoded attachment data, or a URL specifying the location of the attachment data.

Use of inline attachment data is not ideal with CalDAV because the data would need to be uploaded to the server each time a change to the calendar data is done - even minor changes such as a change to the summary. Whilst a client could choose to use a URL value instead, the problem then becomes where and how the client discovers an appropriate URL to use and how it ensures that only those attendees listed in the event or task are able to access it.

This specification solves this problem by having the client send the attachment to the server, separately from the iCalendar data, and the server takes care of adding appropriate "ATTACH" properties in the iCalendar data as well as managing access privileges . The server can also provide additional information to the client about each attachment in the iCalendar data, such as the size and an identifier.

2. Conventions Used in This Document

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

The notation used in this memo is the ABNF notation of [RFC5234] as used by iCalendar [RFC5545]. Any syntax elements shown below that are not explicitly defined in this specification come from iCalendar [RFC5545].

3. Overview

There are four main operations a client needs to do with attachments for calendar data: add, update, remove, and retrieve. The first three operations are carried out by the client issuing an HTTP POST request on the calendar object resource to which the attachment is

associated and specifying the appropriate "action" query parameter (see Section 3.3). In the case of the remove operation, the client can alternatively directly update the calendar object resource and remove the relevant "ATTACH" properties (see Section 3.9). The retrieve operation is accomplished by simply issuing an HTTP GET request targeting the attachment URI specified by the calendar resource's "ATTACH" property (see Section 3.10).

iCalendar data stored in a CalDAV calendar object resource can contain multiple components when recurrences are involved. In such a situation, the client needs to be able to target a specific recurrence instance or multiple instances when adding or deleting attachments. As a result, the POST request needs to provide a way for the client to specify which recurrence instances should be targeted for the attachment operation. This is accomplished through use of additional query parameters on the POST request-URI.

3.1. Requirements

A server that supports the features described in this specification is REQUIRED to support the CalDAV "calendar-access" [RFC4791] features.

In addition, such a server SHOULD support the "return=representation" Prefer header field [RFC7240] preference on successful HTTP PUT and POST requests targeting existing calendar object resources, by returning the new representation of that calendar resource (including its new ETag header field value) in the response.

3.2. Discovering Support for Managed Attachments

A server supporting the features described in this specification MUST include "calendar-managed-attachments" as a field in the DAV response header field from an OPTIONS request on a calendar home collection.

A server might choose to not support storing managed attachments on a per-recurrence instance basis (i.e., they can only be added to all instances as a whole). If that is the case, the server MUST include "calendar-managed-attachments-no-recurrence" as a field in the DAV response header field from an OPTIONS request on a calendar home collection. When that field is present, clients MUST NOT attempt any managed attachment operations that target specific recurrence instances.

3.3. POST Request for Managing Attachments

An HTTP POST request is used to add, update, or remove attachments. The request-URI will contain various query parameters to specify the behavior.

3.3.1. action= Query Parameter

The "action" query parameter is used to identify which attachment operation the client is requesting. This parameter MUST be present once on each POST request used to manage attachments. One of these three values MUST be used:

attachment-add Indicates an operation that is adding an attachment to a calendar object resource. See Section 3.4 for more details.

attachment-update Indicates an operation that is updating an existing attachment on a calendar object resource. See Section 3.5 for more details.

attachment-remove Indicates an operation that is removing an attachment from a calendar object resource. See Section 3.6 for more details.

Example:

http://calendar.example.com/events/1.ics?action=attachment-add

3.3.2. rid= Query Parameter

The "rid" query parameter is used to identify which recurrence instances are being targeted by the client for the attachment operation. This query parameter MUST contain one or more items, separated by commas (0x2C). The item values can be in one of two forms:

Master instance The value "M" (case-insensitive) refers to the "master" recurrence instance, i.e., the component that does not include a "RECURRENCE-ID" property. This item MUST be present only once.

Specific instance A specific iCalendar instance is targeted by using its "RECURRENCE-ID" value as the item value. That value MUST correspond to the RECURRENCE-ID value as stored in the calendar object resource (i.e. without any conversion to UTC). If multiple items of this form are used, they MUST be unique values.

If the "rid" query parameter is not present, all recurrence instances in the calendar object resource are targeted.

The "rid" query parameter MUST NOT be present in the case of an update operation, or if the server chooses not to support per-recurrence instance managed attachments (see Section 3.1).

Example:

```
http://calendar.example.com/events/1.ics?
  action=attachment-add&rid=M,20111022T160000
```

3.3.3. managed-id= Query Parameter

The "managed-id" query parameter is used to identify which "ATTACH" property is being updated or removed. The value of this query parameter MUST match the "MANAGED-ID" property parameter value on the "ATTACH" property in the calendar object resource instance(s) targeted by the request.

The "managed-id" query parameter MUST NOT be present in the case of an add operation.

Example:

```
http://calendar.example.com/events/1.ics?
  action=attachment-update&managed-id=aUNhbGVuZGFy
```

3.4. Adding attachments

To add an attachment to an existing calendar object resource, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.

    A. The request-URI will include an "action" query parameter with the value "attachment-add" (see Section 3.3.1).

    B. If all recurrence instances are having an attachment added, the "rid" query parameter is not present in the request-URI. If one or more specific recurrence instances are targeted, then the request-URI will include a "rid" query parameter containing the list of instances (see Section 3.3.2).

    C. The body of the request contains the data for the attachment.

D. The client MUST include a valid Content-Type header field describing the media type of the attachment (as required by HTTP).

E. The client SHOULD include a Content-Disposition header field [RFC6266] with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment.

F. The client MAY include a Prefer header field [RFC7240] with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.

2. When the server receives the POST request it does the following:

A. Validates that any recurrence instances referred to via the "rid" query parameter are valid for the calendar object resource being targeted.

B. Stores the supplied attachment data into a resource and generates an appropriate URI for clients to access the resource.

C. For each affected recurrence instance in the calendar object resource targeted by the request, the server adds an "ATTACH" property, whose value is the URI of the stored attachment. The "ATTACH" property MUST contain a "MANAGED-ID" parameter whose value is a unique identifier (within the context of the server as a whole). The "ATTACH" property SHOULD contain an "FMTTYPE" parameter whose value matches the Content-Type header field value from the request. The "ATTACH" property SHOULD contain an "FILENAME" parameter whose value matches the Content-Disposition header field "filename" parameter value from the request, taking into account the restrictions expressed in Section 4.2. The "ATTACH" property SHOULD include a "SIZE" parameter whose value represents the size in octets of the attachment. If a specified recurrence instance does not have a matching component in the calendar object resource, then the server MUST modify the calendar object resource to include an overridden component with the appropriate "RECURRENCE-ID" property.

D. Upon successful creation of the attachment resource, and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response and include a "Cal-Managed-ID" header field

containing the "MANAGED-ID" parameter value of the newly created "ATTACH" property. The client can use the "Cal-Managed-ID" header field value to correlate the attachment with "ATTACH" properties added to the calendar object resource. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

In the following example, the client adds a new attachment to a non recurring event and asks the server (via the Prefer [RFC7240] header field) to return the modified version of that event in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition:attachment;filename=agenda.html
Content-Length: xxxx
Prefer: return=representation

<html>
  <body>
    <h1>Agenda</h1>
  </body>
</html>
```

```
   >> Response <<

   HTTP/1.1 201 Created
   Content-Type: text/calendar; charset="utf-8"
   Content-Length: yyyy
   Content-Location: http://cal.example.com/events/64.ics
   ETag: "123456789-000-111"
   Cal-Managed-ID: 97S

   BEGIN:VCALENDAR
   VERSION:2.0
   PRODID:-//Example Corp.//CalDAV Server//EN
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   DTSTAMP:20120201T203412Z
   DTSTART:20120714T170000Z
   DTEND:20120715T040000Z
   SUMMARY:One-off meeting
   ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda.html:https://cal.example.com/attach/64/34X22R
   END:VEVENT
   END:VCALENDAR
```

3.5. Updating Attachments

When an attachment is updated the server MUST change the associated "MANAGED-ID" parameter and MAY change the "ATTACH" property value. With this approach, clients are able to determine when an attachment has been updated by some other client by looking for a change to either the "ATTACH" property value, or the "MANAGED-ID" parameter value.

To change the data of an existing managed attachment in a calendar object resource, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.

    A. The request-URI will include an "action" query parameter with the value "attachment-update" (see Section 3.3.1).

    B. The request-URI will include a "managed-id" query parameter with the value matching that of the "MANAGED-ID" parameter for the "ATTACH" property being updated (see Section 3.3.3).

    C. The body of the request contains the updated data for the attachment.

D. The client MUST include a valid Content-Type header field describing the media type of the attachment (as required by HTTP).

E. The client SHOULD include a Content-Disposition header field [RFC6266] with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment.

F. The client MAY include a Prefer header field [RFC7240] with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.

2. When the server receives the POST request it does the following:

A. Validates that the "managed-id" query parameter is valid for the calendar object resource.

B. Updates the content of the attachment resource corresponding to that managed-id with the supplied attachment data.

C. For each affected recurrence instance in the calendar object resource targeted by the request, the server updates the "ATTACH" property whose "MANAGED-ID" property parameter value matches the "managed-id" query parameter. The "MANAGED-ID" parameter value is changed to allow other clients to detect the update, and the property value (attachment URI) might also be changed. The "ATTACH" property SHOULD contain a "FMTTYPE" parameter whose value matches the Content-Type header field value from the request - this could differ from the original value if the media type of the updated attachment is different. The "ATTACH" property SHOULD contain a "FILENAME" parameter whose value matches the Content-Disposition header field "filename" parameter value from the request, taking into account the restrictions expressed in Section 4.2. The "ATTACH" property SHOULD include a "SIZE" parameter whose value represents the size in octets of the updated attachment.

D. Upon successful update of the attachment resource, and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response, and include a "Cal-Managed-ID" header field containing the new value of the "MANAGED-ID" parameter. The client can use the "Cal-Managed-ID" header field value to correlate the attachment with "ATTACH" properties added to

the calendar object resource. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

The update operation does not take a "rid" parameter and does not add, or remove, any "ATTACH" property in the targeted calendar object resource. To link an existing attachment to a new instance, the client simply does a PUT on the calendar object resource, adding an "ATTACH" property which duplicates the existing one (see Section 3.7).

In the following example, the client updates an existing attachment and asks the server (via the Prefer [RFC7240] header field) to return the updated version of that event in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-update&managed-id=97S HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition:attachment;filename=agenda.html
Content-Length: xxxx
Prefer: return=representation

<html>
  <body>
    <h1>Agenda</h1>
    <p>Discuss attachment draft</p>
  </body>
</html>
```

```
>> Response <<

HTTP/1.1 200 OK
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyz
Content-Location: http://cal.example.com/events/64.ics
Cal-Managed-ID: 98S
ETag: "123456789-000-222"

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
DTSTAMP:20120201T203412Z
DTSTART:20120714T170000Z
DTEND:20120715T040000Z
SUMMARY:One-off meeting
ATTACH;MANAGED-ID=98S;FMTTYPE=text/html;SIZE=xxxy;
 FILENAME=agenda.html:https://cal.example.com/attach/64/34X22R
END:VEVENT
END:VCALENDAR
```

3.6. Removing Attachments via POST

To remove an existing attachment from a calendar object, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.

    A. The request-URI will include an "action" query parameter with the value "attachment-remove" (see Section 3.3.1).

    B. If all recurrence instances are having an attachment removed, the "rid" query parameter is not present in the request-URI. If one or more specific recurrence instances are targeted, then the request-URI will include a "rid" query parameter containing the list of instances (see Section 3.3.2).

    C. The request-URI will include a "managed-id" query parameter with the value matching that of the "MANAGED-ID" property parameter for the "ATTACH" property being removed (see Section 3.3.3).

    D. The body of the request will be empty.

E. The client MAY include a Prefer header field [RFC7240] with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.

2. When the server receives the POST request it does the following:

A. Validates that any recurrence instances referred to via the "rid" query parameter are valid for the calendar object resource being targeted.

B. Validates that the "managed-id" query parameter is valid for the calendar object resource and specific instances being targeted.

C. For each affected recurrence instance in the calendar object resource targeted by the request, the server removes the matching "ATTACH" property. Note that if a specified recurrence instance does not have a matching component in the calendar object resource, then the server MUST modify the calendar object resource to include an overridden component with the appropriate "RECURRENCE-ID" property, and the matching "ATTACH" property removed. This later case is actually valid only if the master component does include the referenced "ATTACH" property.

D. If the attachment resource is no longer referenced by any instance of the calendar object resource, the server can delete the attachment resource to free up storage space.

E. Upon successful removal of the attachment resource and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

In the following example, the client deletes an existing attachment by passing its managed-id in the request. The Prefer [RFC7240] header field is not set in the request so the calendar object resource data is not returned in the response.

```
>> Request <<

POST /events/64.ics?action=attachment-remove&managed-id=98S HTTP/1.1
Host: cal.example.com
Content-Length: 0


>> Response <<


HTTP/1.1 204 No Content
Content-Length: 0
```

3.7. Adding Existing Managed Attachments via PUT

Clients can make use of existing managed attachments by adding the corresponding "ATTACH" property to calendar object resources (subject to the restrictions described in Section 3.12.2). When this occurs, servers SHOULD NOT change either the "MANAGED-ID" parameter value or the "ATTACH" property value for any managed attachments - this ensures that clients do not have to download the attachment data again if they already have it cached, because it is used in another calendar resource.

3.8. Updating Attachments via PUT

Servers MUST NOT allow clients to update attachment data directly via a PUT on the attachment URI (or via any other HTTP method that modifies content). Instead, attachments can only be updated via use of POST requests on the calendar data.

3.9. Removing Attachments via PUT

Clients can remove attachments by simply re-writing the calendar object resource data to remove the appropriate "ATTACH" properties. Servers MUST NOT allow clients to delete attachments directly via a DELETE request on the attachment URI.

3.10. Retrieving Attachments

Clients retrieve attachments by issuing an HTTP GET request using the value of the corresponding "ATTACH" property as the request-URI, taking into account the substitution mechanism associated with the "CALDAV:managed-attachments-server-URL" property (see Section 6.1).

3.11. Error Handling

This specification creates additional preconditions for the POST method.

The new preconditions are:

(CALDAV:max-attachment-size): The attachment submitted in the POST request MUST have an octet size less than or equal to the value of the CALDAV:max-attachment-size property value (Section 6.2) on the calendar collection of the target calendar resource;

(CALDAV:max-attachments-per-resource): The addition of the attachment submitted in the POST request MUST result in the target calendar resource having a number of managed attachments less than or equal to the value of the CALDAV:max-attachments-per-resource property value (Section 6.3) on the calendar collection of the target calendar resource;

(CALDAV:valid-managed-id): The managed-id query parameter in the POST request MUST contain a value corresponding to a "MANAGED-ID" property parameter value in the iCalendar data targeted by the request.

A POST request to add, modify, or delete a managed attachment results in an implicit modification of the targeted calendar resource (equivalent of a PUT). As a consequence, clients should also be prepared to handle preconditions associated with this implicit PUT. This includes (but is not limited to):

(CALDAV:max-resource-size) (from Section 5.3.2.1 of [RFC4791])

(DAV:quota-not-exceeded) (from Section 6 of [RFC4331])

(DAV:sufficient-disk-space) (from Section 6 of [RFC4331])

A PUT request to add or modify and existing calendar object resource can make reference to an existing managed attachment. The following new preconditions is defined:

(CALDAV:valid-managed-id-parameter): a "MANAGED-ID" property parameter value in the iCalendar data in the PUT request is not valid (e.g., does not match any existing managed attachment).

3.12. Additional Considerations

3.12.1. Quotas

The WebDAV Quotas [RFC4331] specification defines two live WebDAV properties (DAV:quota-available-bytes and DAV:quota-used-bytes) to communicate storage quota information to clients. Server implementations MAY choose to include managed attachments sizes when calculating the amount of storage used by a particular resource.

3.12.2. Access Control

Access to the managed attachments store in a calendar object resource SHOULD be restricted to only those calendar users who have access to that calendar object either directly, or indirectly (via being an attendee who would receive a scheduling message).

When accessing a managed attachment, clients SHOULD be prepared to authenticate with the server storing the attachment resource. The credentials required to access the managed attachment store could be different from the ones used to access the CalDAV server.

This specification only allows organizers of scheduled events to add managed attachments. Servers MUST prevent attendees of scheduled events from adding, updating or removing managed attachments. In addition, the server MUST prevent a calendar user from re-using a managed attachment (based on its managed-id value), unless that user is the one who originally created the managed attachment.

3.12.3. Redirects

For POST requests that add or update attachment data, the server MAY issue an HTTP redirect to require the client to re-issue the POST request using a different request-URI. As a result, it is always best for clients to use the "100-continue" expectation defined in Section 5.1.1 of [RFC7231]. Using this mechanism ensures that, if a redirect does occur, the client does not needlessly send the attachment data.

3.12.4. Processing Time

Clients can expect servers to take a while to respond to POST requests that include large attachment bodies. Servers SHOULD use the "102 (Processing)" interim response defined in Section 10.1 of [RFC2518] to keep the client connection alive if the POST request will take significant time to complete.

3.12.5. Automatic Clean-Up by Servers

Servers MAY automatically remove attachment data, for example to regain the storage taken by unused attachments, or as the result of a virus scanning. When doing so they SHOULD NOT modify calendar data referencing those attachments. Instead they SHOULD respond with "410 (Gone)" to any request on the removed attachment URI.

3.12.6. Sending Scheduling Messages with Attachments

When a managed attachment is added, updated or removed from a calendar object resource, the server MUST ensure that a scheduling message is sent to update any attendees with the changes, as per [RFC6638].

3.12.7. Migrating Calendar Data

When exporting calendar data from a CalDAV server supporting managed attachments, clients SHOULD remove all "MANAGED-ID" property parameters from "ATTACH" properties in the calendar data. Similarly when importing calendar data from another source, clients SHOULD remove any "MANAGED-ID" property parameters on "ATTACH" properties (failure to do so will likely result in the server removing those properties automatically).

4. Modifications to iCalendar Syntax

4.1. SIZE Property Parameter

Parameter Name: SIZE

Purpose: To specify the size of an attachment.

Format Definition: This property parameter is defined by the following notation:

```
sizeparam = "SIZE" "=" paramtext
; positive integers
```

Description: This property parameter MAY be specified on "ATTACH" properties. It indicates the size in octets of the corresponding attachment data. Since iCalendar integer values are restricted to a maximum value of 2147483647, the current parameter is defined as text to allow an extended range to be used.

Example:

```
ATTACH;SIZE=1234:http://attachments.example.com/abcd.txt
```

4.2. FILENAME Property Parameter

Parameter Name: FILENAME

Purpose: To specify the file name of a managed attachment.

Format Definition: This property parameter is defined by the following notation:

filenameparam = "FILENAME" "=" paramtext

Description: This property parameter MAY be specified on "ATTACH" properties corresponding to managed attachments. Its value provides information on how to construct a filename for storing the attachment data. This parameter is very similar in nature to the Content-Disposition HTTP header field "filename" parameter and exposes the same security risks. As a consequence, clients MUST follow the guidelines expressed in Section 4.3 of [RFC6266] when consuming this parameter value. Similarly, servers MUST follow those same guidelines before storing a value.

Example:

```
ATTACH;FILENAME=agenda.html:http://attachments.example.c
 om/rt452S
```

4.3. MANAGED-ID Property Parameter

Parameter Name: MANAGED-ID

Purpose: To uniquely identify a managed attachment.

Format Definition: This property parameter is defined by the following notation:

managedidparam = "MANAGED-ID" "=" paramtext

Description: This property parameter MUST be specified on "ATTACH" properties corresponding to managed attachments. Its value is generated by the server and uniquely identifies a managed attachment. This property parameter MUST NOT be present in the case of non-managed attachments.

Example:

```
ATTACH;MANAGED-ID=aUNhbGVuZGFy:http://attachments.example.c
 om/abcd.txt
```

5. Additional Message Header Fields

5.1. Cal-Managed-ID Response Header Field

The Cal-Managed-ID response header field provides the value of the MANAGED-ID parameter corresponding to a newly added ATTACH property. It MUST be sent only in response to a successful POST request with an action set to attachment-add or attachment-update.

```
Cal-Managed-ID = "Cal-Managed-ID" ":" paramtext
; "paramtext" is defined in Section 3.1 of [RFC5545]
```

Example:

```
Cal-Managed-ID:aUNhbGVuZGFy
```

6. Additional WebDAV Properties

6.1. CALDAV:managed-attachments-server-URL property

Name: managed-attachments-server-URL

Namespace: urn:ietf:params:xml:ns:caldav

Purpose: Specifies the server base URI to use when retrieving managed attachments.

Protected: This property MUST be protected as only the server can update the value.

COPY/MOVE behavior: This property is only defined on a calendar home collection which cannot be moved or copied.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: This property MAY be defined on a calendar home collection. If present, it contains zero or one DAV:href XML elements.

When one DAV:href element is present, its value MUST be an absolute HTTP URI containing only the scheme (i.e. "https") and authority (i.e. host and port) parts . Whenever a managed attachment is to be retrieved via an HTTP GET, the client MUST construct the actual URL of the attachment by substituting the scheme and authority parts of the attachment URI (as stored in the iCalendar "ATTACH" property) with the present WebDAV property value.

When no DAV:href element is present, the client MUST substitute the scheme and authority parts of the attachment URI with the scheme and authority part of the calendar home collection absolute URI.

In the absence of this property, the client can consider the attachment URI as its actual URL.

Definition:

```
<!ELEMENT managed-attachments-server-URL (DAV:href?)>
```

Example:

```
<C:managed-attachments-server-URL xmlns:D="DAV:"
    xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:href>https://attachstore.example.com</D:href>
</C:managed-attachments-server-URL>
```

6.2. CALDAV:max-attachment-size property

Name: max-attachment-size

Namespace: urn:ietf:params:xml:ns:caldav

Purpose: Provides a numeric value indicating the maximum attachment size, in octets, that the server is willing to accept when a managed attachment is stored on the server.

Protected: MUST be protected as it indicates limits provided by the server.

COPY/MOVE behavior: This property value MUST be preserved in COPY and MOVE operations.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: The CALDAV:max-attachment-size property is used to specify a numeric value that represents the maximum attachment size, in octets, that the server is willing to accept when a managed attachment is stored on the server. The property is defined on the parent collection of the calendar object resource to which the attachment is associated. Any attempt to store a managed attachment exceeding this size MUST result in an error, with the CALDAV:max-attachment-size precondition (Section 3.11) being violated. In the absence of this property, the client can

assume that the server will allow storing an attachment of any reasonable size.

Definition:

```
<!ELEMENT max-attachment-size (#PCDATA)>
<!-- PCDATA value: a numeric value (positive decimal integer) -->
```

Example:

```
<C:max-attachment-size xmlns:C="urn:ietf:params:xml:ns:caldav"
     >102400000</C:max-attachment-size>
```

6.3. CALDAV:max-attachments-per-resource property

Name: max-attachments-per-resource

Namespace: urn:ietf:params:xml:ns:caldav

Purpose: Provides a numeric value indicating the maximum number of managed attachments across all instances of a calendar object resource stored in a calendar collection.

Protected: MUST be protected as it indicates limits provided by the server.

COPY/MOVE behavior: This property value MUST be preserved in COPY and MOVE operations.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: The CALDAV:max-attachments-per-resource property is used to specify a numeric value that represents the maximum number of managed attachments across all instances of a calendar object resource stored in a calendar collection. Non-managed attachments are not counted toward that limit. The property is defined on the parent collection of the calendar object resource to which the attachment is associated. Any attempt to add a managed attachment that would cause the calendar resource to exceed this limit MUST result in an error, with the CALDAV:max-attachments-per-resource precondition (Section 3.11) being violated. In the absence of this property, the client can assume that the server can handle any number of managed attachments per calendar resource.

Definition:

```
<!ELEMENT max-attachments-per-resource (#PCDATA)>
<!-- PCDATA value: a numeric value (positive decimal integer) -->
```

Example:

```
<C:max-attachments-per-resource
    xmlns:C="urn:ietf:params:xml:ns:caldav"
     >12</C:max-attachments-per-resource>
```

7. Implementation Status

< RFC Editor: before publication please remove this section and the reference to [RFC7942] >

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. Calendar and Contacts Server

The open source Calendar and Contacts Server [1] project is a standards-compliant server implementing the CalDAV protocol. This production level implementation supports all of the requirements described in this document and successfully interoperates with the Apple Calendar, BusyCal, 2Do, and CalDAVTester client implementations described below. This implementation is freely distributable under the terms of the Apache License, Version 2.0 [2].

7.2. Cyrus Server

The open source Cyrus Server [3] project is a highly scalable enterprise mail system which also supports calendaring. This production level CalDAV implementation supports all of the requirements described in this document and successfully interoperates with the Apple Calendar and CalDAVTester client implementations described below. This implementation is freely distributable under a BSD style license from Computing Services at Carnegie Mellon University [4].

7.3. Oracle Communications Calendar Server

The Oracle Communications Calendar Server [5] project is a standards-compliant, scalable, enterprise-ready calendaring solution. This production level CalDAV implementation supports all of the requirements described in this document and successfully interoperates with the Apple Calendar and CalDAVTester client implementations described below. This implementation is proprietary and available for a free trial and/or purchase from the vendor.

7.4. Apple Calendar

The widely used Apple Calendar [6] client is a standards-compliant client implementing the CalDAV protocol. This production level implementation supports all the requirements described in this document and successfully interoperates with the Calendar and Contacts Server, Cyrus Server, and Oracle Communications Calendar Server implementations described above. This client implementation is proprietary and is distributed as part of Apple’s desktop operating systems.

7.5. BusyCal

BusyCal [7] is a standards-compliant calendar client for MacOS implementing the CalDAV protocol. This implementation supports all of the requirements described in this document and successfully interoperates with the Calendar and Contacts Server and Cyrus Server implementations described above. This implementation is proprietary and available for a free trial and/or purchase from the vendor.

7.6. CalDAVTester

CalDAVTester [8] is an open source test and performance application designed to work with CalDAV servers and tests various aspects of their protocol handling as well as performance. This widely used implementation supports all of the requirements described in this document and successfully interoperates with the server

implementations described above. This implementation is freely distributable under the terms of the Apache License, Version 2.0 [9].

7.7. 2Do

2Do [10] is a standards-complient calendar client for iOS which uses the CalDAV standard for communication. This implementation supports all of the requirements described in this document and successfully interoperates with the Calendar and Contacts Server implementation described above. This implementation is proprietary and available for purchase from the vendor.

8. Security Considerations

Malicious content could be introduced into the Calendar Server by way of a managed attachment, and propagated to many end users via scheduling. Servers SHOULD check managed attachments for malicious or inappropriate content. Upon detecting of such content, servers SHOULD remove the attachment, following the rules described in Section 3.12.5.

9. IANA Considerations

9.1. Parameter Registrations

This specification defines the following new iCalendar property parameters to be added to the registry defined in Section 8.2.3 of [RFC5545]:

Property Parameter	Status	Reference
SIZE	Current	RFCXXXX, Section 4.1
FILENAME	Current	RFCXXXX, Section 4.2
MANAGED-ID	Current	RFCXXXX, Section 4.3

9.2. Message Header Field Registrations

The message header fields below should be added to the Permanent Message Header Field Registry (see [RFC3864]).

9.2.1. Cal-Managed-ID

Header field name: Cal-Managed-ID

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification (Section 5.1)

Related information: none

10. Acknowledgments

This specification came about via discussions at the Calendaring and Scheduling Consortium. Thanks in particular to Mike Douglass and Eric York.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and D. Jensen, "HTTP Extensions for Distributed Authoring -- WEBDAV", RFC 2518, DOI 10.17487/RFC2518, February 1999, <http://www.rfc-editor.org/info/rfc2518>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <http://www.rfc-editor.org/info/rfc3864>.

[RFC4331] Korver, B. and L. Dusseault, "Quota and Size Properties for Distributed Authoring and Versioning (DAV) Collections", RFC 4331, DOI 10.17487/RFC4331, February 2006, <http://www.rfc-editor.org/info/rfc4331>.

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <http://www.rfc-editor.org/info/rfc4791>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <http://www.rfc-editor.org/info/rfc5545>.

[RFC6266] Reschke, J., "Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)", RFC 6266, DOI 10.17487/RFC6266, June 2011, <http://www.rfc-editor.org/info/rfc6266>.

[RFC6638] Daboo, C. and B. Desruisseaux, "Scheduling Extensions to CalDAV", RFC 6638, DOI 10.17487/RFC6638, June 2012, <http://www.rfc-editor.org/info/rfc6638>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7240] Snell, J., "Prefer Header for HTTP", RFC 7240, DOI 10.17487/RFC7240, June 2014, <http://www.rfc-editor.org/info/rfc7240>.

11.2. Informative References

[RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <http://www.rfc-editor.org/info/rfc5546>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <http://www.rfc-editor.org/info/rfc7942>.

[RFC8144] Murchison, K., "Use of the Prefer Header Field in Web Distributed Authoring and Versioning (WebDAV)", RFC 8144, DOI 10.17487/RFC8144, April 2017, <http://www.rfc-editor.org/info/rfc8144>.

11.3. URIs

[1] http://calendarserver.org/

[2] http://www.apache.org/licenses/LICENSE-2.0.html

[3] http://www.cyrusimap.org/

[4] http://www.cmu.edu/computing/

[5] http://www.cyrusimap.org/

[6] http://www.apple.com/macos/

[7] http://www.busymac.com/busycal/

[8] http://calendarserver.org/wiki/CalDAVTester

[9] http://www.apache.org/licenses/LICENSE-2.0.html

[10] http://www.2doapp.com/

Appendix A. Change History (To be removed by RFC Editor before publication)

Changes in calext-03:

1. Changed to Informational based on feedback regarding non-standard method of updating an existing resource.

2. Added references to sub-sections in Overview.

3. Made support for Prefer header field a SHOULD for servers.

4. Expanded recurring event examples to use conditional requests and to include the Expect header field.

5. Minor editorial changes.

Changes in calext-02:

1. Moved "Error Handling" into its own sub-section.

2. Split "Other Client Considerations" into "Processing Time" and "Migrating Calendar Data".

Changes in calext-01:

1. Changed all instances of "header" to "header field".

2. Reworked wording of Prefer header field handling.

3. Switched to recommending 102 (Processing) interim response to keep the client connection alive.

4. Fixed description of Cal-Managed-ID response header field to state that it is also required in responses to successful attachment-update.

5. Minor editorial changes.

Changes in calext-00:

1. Added Murchison as editor.

2. Updated HTTP references to RFC7230 and RFC7231.

3. Updated Prefer header field references to RFC7240.

4. Added Implementation Status section.

5. Minor editorial changes.

Changes in daboo-03:

1. Fixed some examples.

2. Fixed return-representation -> return=representation.

3. Added statement that servers must not allow clients to DELETE attachments directly.

4. Added new preconditions for valid managed-id values.

5. Filled out Access Control section.

6. Allow servers to not support per-instance attachments and advertise that fact to clients.

Changes in daboo-02:

1. MANAGED-ID changes on PUT.

2. MTAG has been removed.

3. Error pre-conditions added.

4. Interaction with WebDAV QUOTA discussed.

5. max-attachment-* limits added.

6. Updated references.

7. Removed MUST for specific 2xx codes in favor of generic success code.

Changes in daboo-01:

1. Tweaked OPTIONS capability wording.

2. Added section on clients expecting 100-Continue for delayed response.

3. Added text for clean-up and use of HTTP 410 on orphans.

4. Added text on removing "MANAGED-ID" when exporting/importing calendar data.

5. Added protocol examples.

6. Added MTAG property parameter on ATTACH property

7. Added FILENAME property parameter on ATTACH property

8. "id" query parameter is now "managed-id".

9. Use of Cal-Managed-ID header instead of Location header in responses.

10. rid query param MUST contain RECURRENCE-ID without any conversion to UTC (case of floating events).

11. Introduced CALDAV:managed-attachments-server-URL property

12. Made support for Prefer header a MUST for servers.

Appendix B. Example Involving Recurring Events

In the following example, the organizer of a recurring meeting makes an unsuccessful attempt to add an agenda (HTML attachment) to the corresponding calendar resource with a conditional request. Note that the client includes both the Expect and Prefer header fields in the request, thereby preventing itself from needlessly sending the attachment data, and requesting that the current resource be returned in the failure response (see Section 3.2 of [RFC8144]).

```
   >> Request <<

   POST /events/65.ics?action=attachment-add HTTP/1.1
   Host: cal.example.com
   Content-Type: text/html; charset="utf-8"
   Content-Disposition: attachment;filename=agenda.html
   Content-Length: xxxx
   If-Match: "abcdefg-000"
   Expect: 100-continue
   Prefer: return=representation
```

```
   >> Final Response <<

   HTTP/1.1 412 Precondition Failed
   Content-Type: text/calendar; charset="utf-8"
   Content-Length: yyyy
   Content-Location: http://cal.example.com/events/65.ics
   ETag: "123456789-000-000"

   BEGIN:VCALENDAR
   VERSION:2.0
   PRODID:-//Example Corp.//CalDAV Server//EN
   BEGIN:VTIMEZONE
   LAST-MODIFIED:20040110T032845Z
   TZID:America/Montreal
   BEGIN:DAYLIGHT
   DTSTART:20000404T020000
   RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
   TZNAME:EDT
   TZOFFSETFROM:-0500
   TZOFFSETTO:-0400
   END:DAYLIGHT
   BEGIN:STANDARD
   DTSTART:20001026T020000
   RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
   TZNAME:EST
   TZOFFSETFROM:-0400
   TZOFFSETTO:-0500
   END:STANDARD
   END:VTIMEZONE
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120206T100000
   DURATION:PT1H
   RRULE:FREQ=WEEKLY
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exam
    ple.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@exa
    mple.com
   END:VEVENT
   END:VCALENDAR
```

The organizer of a recurring meeting successfully adds an agenda (HTML attachment) to the corresponding calendar resource. Attendees

of the meeting are granted read access to the newly created attachment resource. Their own copy of the meeting is updated to include the new ATTACH property pointing to the attachment resource and they are notified of the change via their scheduling inbox.

```
>> Request <<

POST /events/65.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment;filename=agenda.html
Content-Length: xxxx
If-Match: "123456789-000-000"
Expect: 100-continue
Prefer: return=representation


>> Interim Response <<

HTTP/1.1 100 Continue


>> Request Body <<

<html>
  <body>
    <h1>Agenda</h1>
    <p>As usual</p>
  </body>
</html>
```

```
   >> Final Response <<

   HTTP/1.1 201 Created
   Content-Type: text/calendar; charset="utf-8"
   Content-Length: yyyy
   Content-Location: http://cal.example.com/events/65.ics
   ETag: "123456789-000-111"
   Cal-Managed-ID: 97S

   BEGIN:VCALENDAR
   VERSION:2.0
   PRODID:-//Example Corp.//CalDAV Server//EN
   BEGIN:VTIMEZONE
   LAST-MODIFIED:20040110T032845Z
   TZID:America/Montreal
   BEGIN:DAYLIGHT
   DTSTART:20000404T020000
   RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
   TZNAME:EDT
   TZOFFSETFROM:-0500
   TZOFFSETTO:-0400
   END:DAYLIGHT
   BEGIN:STANDARD
   DTSTART:20001026T020000
   RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
   TZNAME:EST
   TZOFFSETFROM:-0400
   TZOFFSETTO:-0500
   END:STANDARD
   END:VTIMEZONE
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120206T100000
   DURATION:PT1H
   RRULE:FREQ=WEEKLY
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exam
    ple.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@exa
    mple.com
   ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda.html:https://cal.example.com/attach/65/34X22R
   END:VEVENT
   END:VCALENDAR
```

The organizer has a more specific agenda for the 20th of February meeting. It is added to that particular instance of the meeting by specifying the rid parameter. Note that the server takes significant time to complete the request and notifies the client accordingly.

>> Request <<

```
POST /events/65.ics?action=attachment-add&rid=20120220T100000 HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment;filename=agenda0220.html
Content-Length: xxxx
If-Match: "123456789-000-111"
Expect: 100-continue
Prefer: return=representation
```

>> Interim Response <<

```
HTTP/1.1 100 Continue
```

>> Request Body <<

```
<html>
  <body>
    <h1>Agenda</h1>
    <p>Something different, for a change</p>
  </body>
</html>
```

>> Interim Response <<

```
HTTP/1.1 102 Processing
```

>> Final Response <<

```
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: http://cal.example.com/events/65.ics
ETag: "123456789-000-222"
Cal-Managed-ID: 33225

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
```

```
   BEGIN:VTIMEZONE
   LAST-MODIFIED:20040110T032845Z
   TZID:America/Montreal
   BEGIN:DAYLIGHT
   DTSTART:20000404T020000
   RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
   TZNAME:EDT
   TZOFFSETFROM:-0500
   TZOFFSETTO:-0400
   END:DAYLIGHT
   BEGIN:STANDARD
   DTSTART:20001026T020000
   RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
   TZNAME:EST
   TZOFFSETFROM:-0400
   TZOFFSETTO:-0500
   END:STANDARD
   END:VTIMEZONE
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120206T100000
   DURATION:PT1H
   RRULE:FREQ=WEEKLY
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exam
    ple.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@exa
    mple.com
   ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda.html:https://cal.example.com/attach/65/34X22R
   END:VEVENT
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   RECURRENCE-ID;TZID=America/Montreal:20120220T100000
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120220T100000
   DURATION:PT1H
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@example.
    com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@examp
```

```
    le.com
   ATTACH;MANAGED-ID=33225;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda0220.html:https://cal.example.com/attach/65/FGZ225
   END:VEVENT
   END:VCALENDAR
```

Authors' Addresses

Cyrus Daboo
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Email: cyrus@daboo.name
URI: http://www.apple.com/

Arnaud Quillaud
Oracle Corporation
180, Avenue de l'Europe
Saint Ismier cedex 38334
France

Email: arnaud.quillaud@oracle.com
URI: http://www.oracle.com/

Kenneth Murchison (editor)
FastMail Pty Ltd.
Level 1, 91 William Street
Melbourne, VIC 3000
Australia

Email: murch@fastmail.com
URI: http://www.fastmail.com/

Network Working Group
Internet-Draft
Updates: 5545,5546 (if approved)
Intended status: Standards Track
Expires: April 14, 2018

M. Douglass
Spherical Cow Group

October 11, 2017

Event Publishing Extensions to iCalendar
draft-ietf-calext-eventpub-extensions-05

Abstract

This specification introduces a number of new iCalendar properties and components which are of particular use for event publishers and in social networking.

This specification also defines a new STRUCTURED-DATA property for iCalendar [RFC5545] to allow for data that is directly pertinent to an event or task to be included with the calendar data.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 14, 2018.

Copyright Notice

Table of Contents

1. Introduction

The currently existing iCalendar standard [RFC5545] lacks useful methods for referencing additional, external information relating to calendar components. Additionally there is no standard way to provide rich text descriptions or meta-data associated with the event.

Current practice is to embed this information as links in the description or to add x-properties.

This document defines a number of properties and a component referencing such external information that can provide additional information about an iCalendar component. The intent is to allow interchange of such information between applications or systems (e.g., between clients, between client and server, and between servers). Formats such as VCARD are likely to be most useful to the receivers of such events as they may be used in other applications - such as address books.

This specification defines a new PARTICIPANT component. Many people or groups may participate in an event. This component provides detailed information. Such participants may act as attendees to the event (or derived events) or may just provide a reference - perhaps for mailing lists.

The following properties are defined in this specification

STYLED-DESCRIPTION: Supports HTML descriptions. Event publishers typically wish to provide more and better formatted information about the event.

STRUCTURED-LOCATION: There may be a number of locations associated with an event. This provides detailed information about the location.

STRUCTURED-RESOURCE: Events need resources such as rooms, projectors, conferencing capabilities.

STRUCTURED-DATA: The existing properties in iCalendar cover key elements of events and tasks such as start time, end time, location, summary, etc. However, different types of events often have other specific "fields" that it is useful to include in the calendar data. For example, an event representing an airline flight could include the airline, flight number, departure and

arrival airport codes, check-in and gate-closing times etc. As another example, a sporting event might contain information about the type of sport, the home and away teams, the league the teams are in, information about nearby parking, etc.

PARTICIPANT-TYPE: Used in the PARTICIPANT component to define the type.

CALENDAR-ADDRESS: Used in the PARTICIPANT component to provide the calendar address of the participant.

In addition the SOURCE property defined in [RFC7986] is redefined to allow VALUE=TEXT and broaden its usage.

1.1. Conventions Used in This Document

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

2. Components and properties

Previous extensions to the calendaring standards have been largely restricted to the addition of properties or parameters. This is partly because iCalendar libraries had trouble handling components nested deeper than those defined in [RFC5545]

In a break with this ’tradition’ this specification introduces one of these extensions as a component rather than a property. This is a better match for the way XML and JSON handles such structures and allows richer definitions.

It also allows for the addition of extra properties inside the component and resolves some of the problems of trying to add detailed information as a parameter.

3. Typed References

The properties defined here can all reference external meta-data which may be used by applications to provide enhanced value to users. By providing type information as parameters, clients and servers are able to discover interesting references and make use of them, perhaps for indexing or the presentation of additional related information for the user.

The [RFC5545] LOCATION property provides only an unstructured single text value for specifying the location where an event (or task) will

occur. This is inadequate for use cases where structured location information (e.g. address, region, country, postal code) is required or preferred, and limits widespread adoption of iCalendar in those settings.

Using STRUCTURED-LOCATION, information about a number of interesting locations can be communicated, for example, parking, restaurants and the venue. Servers and clients can retrieve the objects when storing the event and use them to index by geographic location.

When a calendar client receives a calendar component it can search the set of supplied properties looking for those of particular interest. The TYPE and FMTTYPE parameters, if supplied, can be used to help the selection.

The PARTICIPANT component is designed to handle common use cases in event publication. It is generally important to provide information about the organizers of such events. Sponsors wish to be referenced in a prominent manner. In social calendaring it is often important to identify the active participants in the event, for example a school sports team, and the inactive participants, for example the parents.

The PARTICIPANT component canalso be used to provide useful extra daat about an attendee. For example a LOCATION property inside the PARTICIPANT gives the actual location of a remote attendee.

3.1. Use Cases

The main motivation for these properties has been event publication but there are opportunities for use elsewhere. The following use cases will describe some possible scenarios.

3.1.1. Piano Concert Performance

In putting together a concert there are many participants: piano tuner, performer, stage hands etc. In addition there are sponsors and various contacts to be provided. There will also be a number of related locations. A number of events can be created, all of which relate to the performance in different ways.

There may be an iTip [RFC5546] meeting request for the piano tuner who will arrive before the performance. Other members of staff may also receive meeting requests.

An event can also be created for publication which will have a PARTICIPANT component for the pianist providing a reference to vcard information about the performer. This event would also hold

information about parking, local subway stations and the venue itself. In addition, there will be sponsorship information for sponsors of the event and perhaps paid sponsorship properties essentially advertising local establishments.

3.1.2. Itineraries

These additions also provide opportunities for the travel industry. When booking a flight the PARTICIPANT component can be used to provide references to businesses at the airports and to car hire businesses at the destination.

The embedded location information can guide the traveller at the airport or to their final destination. The contact information can provide detailed information about the booking agent, the airlines and car hire companies and the hotel.

4. Modifications to Calendar Components

The following changes to the syntax defined in iCalendar [RFC5545] are made here. New elements are defined in subsequent sections.

```
   eventc     = "BEGIN" ":" "VEVENT" CRLF
                eventprop *alarmc *participantc
                "END" ":" "VEVENT" CRLF

   eventprop =/ *(
                ;
                ; The following are OPTIONAL,
                ; and MAY occur more than once.
                ;
                styleddescription / strucloc / strucres / sdataprop
                ;
                )

   todoc      = "BEGIN" ":" "VTODO" CRLF
                todoprop *alarmc *participantc
                "END" ":" "VTODO" CRLF

   todoprop =/ *(
                ;
                ; The following are OPTIONAL,
                ; and MAY occur more than once.
                ;
                styleddescription / strucloc / strucres / sdataprop
                ;
                )

   journalc   = "BEGIN" ":" "VJOURNAL" CRLF
                jourprop *participantc
                "END" ":" "VJOURNAL" CRLF


   jourprop =/ *(
                ;
                ; The following are OPTIONAL,
                ; and MAY occur more than once.
                ;
                styleddescription / sdataprop
                ;
                )
```

5. New Property Parameters

This specification makes use of the LABEL property parameter which is defined in [RFC7986]

5.1. Loctype

Parameter name: LOCTYPE

Purpose: To specify the type of location.

Format Definition:

This parameter is defined by the following notation:

```
  loctypeparam   = "LOCTYPE" "=" param-value
```

Description: This parameter MAY be specified on STRUCTURED-LOCATION and provides a way to differentiate multiple properties. For example, it allows event producers to provide location information for the venue and the parking.

Values for this parameter are taken from the values defined in [RFC4589]. New location types SHOULD be registered in the manner laid down in that specification

5.2. Restype

Parameter name: RESTYPE

Purpose: To specify the type of resource.

Format Definition:

This parameter is defined by the following notation:

```
  restypeparam   = "RESTYPE" "=" param-value
```

Description: This parameter MAY be specified on STRUCTURED-RESOURCE and provides a way to differentiate multiple properties.

The allowable values are defined in Section 10 New resource types SHOULD be registered in the manner laid down in this specification

5.3. Order

Parameter name: ORDER

Purpose: To define ordering for the associated property.

Format Definition:

This parameter is defined by the following notation:

```
orderparam    = "ORDER" "=" integer ;Must be greater than or equal to 1
```

Description: The ORDER parameter is OPTIONAL and is used to indicate the relative ordering of the corresponding instance of a property. Its value MUST be an integer greater than or equal to 1 that quantifies the order with 1 being the first in the ordering.

When the parameter is absent, the default MUST be to interpret the property instance as being at the lowest level of ordering, that is, the property will appear after any other instances of the same property with any value of ORDER.

Note that the value of this parameter is to be interpreted only in relation to values assigned to other corresponding instances of the same property in the same entity. A given value, or the absence of a value, MUST NOT be interpreted on its own.

This parameter MAY be applied to any property that allows multiple instances.

5.4. Schema

Parameter Name: SCHEMA

Purpose: To specify the schema used for the content of a "STRUCTURED-DATA" property value.

Format Definition:

This parameter is defined by the following notation:

```
schemaparam    = "SCHEMA" "=" DQUOTE uri DQUOTE
```

Description: This property parameter SHOULD be specified on "STRUCTURED-DATA" properties. When present it provides identifying information about the nature of the content of the corresponding "STRUCTURED-DATA" property value. This can be used to supplement the media type information provided by the "FMTTYPE" parameter on the corresponding property.

Example:

```
STRUCTURED-DATA;FMTTYPE=application/ld+json;
 SCHEMA="https://schema.org/FlightReservation";
 ENCODING=BASE64;VALUE=BINARY:Zm9vYmFy
```

6. Redefined Property SOURCE

The SOURCE property defined in [RFC7986] is redefined to allow VALUE=TEXT and broaden its usage to any component.

Property name: SOURCE

Purpose: This property provides a reference to information about a component such as a participant possibly as a vcard or optionally a plain text typed value.

Value type: The default value type for this property is URI. The value type can also be set to TEXT to indicate plain text content.

Property Parameters: Non-standard or format type parameters can be specified on this property.

Conformance: This property MAY be appear in any iCalendar component.

Description: This property provides information about the component in which it appears.

In a resource or participant it may provide a reference to a vcard giving directory information.

In a VCALENDAR component this property identifies a location where a client can retrieve updated data for the calendar. Clients SHOULD honor any specified "REFRESH-INTERVAL" value when periodically retrieving data. Note that this property differs from the "URL" property in that "URL" is meant to provide an alternative representation of the calendar data rather than the original location of the data.

In a calendar entity component such as an event the SOURCE property may provide a reference to the original source of the event. This may be used by aggregators to provide a link back.

Format Definition:

This property is defined by the following notation:

```
    source     = "SOURCE" sourceparam
                 (
                   (
                     ";" "VALUE" "=" "URI"
                     ":" uri
                   )  /
                   (
                     ";" "VALUE" "=" "TEXT"
                     ":" text
                   )
                 )
                 CRLF

    sourceparam      = *(
                 ;
                 ; the following are OPTIONAL
                 ; but MUST NOT occur more than once
                 ;
                 (";" fmttypeparam) /
                 ;
                 ; the following is OPTIONAL
                 ; and MAY occur more than once
                 ;
                 (";" other-param)
                 ;
                 )
```

Example:

The following is an example referring to a VCARD.

```
    SOURCE;FMTTYPE=text/vcard;VALUE=URL:
     http://dir.example.com/vcard/contacts/contact1.vcf
```

7. New Properties

7.1. Participant Type

Property name: PARTICIPANT-TYPE

Purpose: To specify the type of participant.

Value type: The value type for this property is TEXT. The allowable values are defined in Section 9.

Property Parameters: Non-standard parameters can be specified on this property.

Conformance: This property MUST be specified within a PARTICIPANT component.

Description: This property defines the type of participation in events or tasks. Participants can be individuals or organizations, for example a soccer team, the spectators, or the musicians.

Format Definition:

This parameter is defined by the following notation:

```
participanttype   = "PARTICIPANT-TYPE" "=" iana-token
```

7.2. Calendar Address

Property name: CALENDAR-ADDRESS

Purpose: To specify the calendar address for a participant.

Value type: CAL-ADDRESS

Property Parameters: IANA or non-standard property parameters can be specified on this property.

Conformance: This property MAY be specified within a PARTICIPANT component.

Description: This property provides a calendar user address for the participant. If there is an ATTENDEE property with the same value then the participant is schedulable.

Format Definition:

This parameter is defined by the following notation:

```
calendaraddress   = "CALENDAR-ADDRESS" "=" cal-address
```

7.3. Styled-Description

Property name: STYLED-DESCRIPTION

Purpose: This property provides for one or more rich-text descriptions to replace or augment that provided by the DESCRIPTION property.

Value type: There is no default value type for this property. The value type can be set to URI or TEXT. Other text-based value types can be used when defined in the future. Clients MUST ignore any properties with value types they do not understand.

Property Parameters: IANA, non-standard, id, alternate text representation, format type, and language property parameters can be specified on this property.

Conformance: The property can be specified multiple times in the "VEVENT", "VTODO", "VJOURNAL", or "VALARM" calendar components.

Description: This property is used in the "VEVENT" and "VTODO" to capture lengthy textual descriptions associated with the activity. This property is used in the "VJOURNAL" calendar component to capture one or more textual journal entries. This property is used in the "VALARM" calendar component to capture the display text for a DISPLAY category of alarm, and to capture the body text for an EMAIL category of alarm.

VALUE=TEXT is used to provide rich-text variants of the plain-text DESCRIPTION property.

VALUE=URI is used to provide a link to rich-text content which is expected to be displayed inline as part of the event.

The intent of this property is limited to providing a styled and/or language specific version of the DESCRIPTION property. The URL property should be used to link to websites or other related information.

Applications MAY attempt to guess the media type of the resource via inspection of its content if and only if the media type of the resource is not given by the "FMTTYPE" parameter. If the media type remains unknown, calendar applications SHOULD treat it as type "text/html".

Multiple STYLED-DESCRIPTION properties may be used to provide different formats or different language variants.

Format Definition:

This property is defined by the following notation:

```
  styleddescription = "STYLED-DESCRIPTION" styleddescparam ":"
                      (
                        (
                          ";" "VALUE" "=" "URI"
                          ":" uri
                        )  /
                        (
                          ";" "VALUE" "=" "TEXT"
                          ":" text
                        )
                      )
                      CRLF

  styleddescparam   = *(
                    ;
                    ; The following are OPTIONAL,
                    ; but MUST NOT occur more than once.
                    ;
                    (";" altrepparam) / (";" languageparam) /
                    (";" fmttypeparam) /
                    ;
                    ; the following is OPTIONAL
                    ; and MAY occur more than once
                    ;
                    (";" other-param)
                    )
```

Example:

The following is an example of this property. It points to an html description.

```
       STYLED-DESCRIPTION;VALUE=URI:http://example.org/desc001.html
```

7.4. Structured-Location

Property name: STRUCTURED-LOCATION

Purpose: This property provides a typed reference to external information about the location of an event or optionally a plain text typed value.

Value type: There is no default value type for this property. The value type can be set to URI or TEXT.

Property Parameters: IANA, non-standard, label, loctype or format type parameters can be specified on this property.

Conformance: This property MAY be specified zero or more times in any iCalendar component.

Description: When used in a component the value of this property provides information about the event venue or of related services such as parking, dining, stations etc..

When a LABEL parameter is supplied the language of the label must match that of the content and of the LANGUAGE parameter if present.

Format Definition:

This property is defined by the following notation:

```
  strucloc        = "STRUCTURED-LOCATION" struclocparam
                    (
                      (
                        ";" "VALUE" "=" "URI"
                        ":" uri
                      )  /
                      (
                        ";" "VALUE" "=" "TEXT"
                        ":" text
                      )
                    )
                    CRLF

  struclocparam   = *(
                  ;
                  ; the following are OPTIONAL
                  ; but MUST NOT occur more than once
                  ;
                  (";" fmttypeparam) /
                  (";" labelparam) /
                  (";" languageparam) /
                  (";" loctypeparam) /
                  ;
                  ; the following is OPTIONAL
                  ; and MAY occur more than once
                  ;
                  (";" other-param)
                  )
```

Example:

The following is an example of this property. It points to a venue.

```
STRUCTURED-LOCATION;LABEL="The venue":
  http://dir.example.com/venues/big-hall.vcf
```

7.5. Structured-Resource

Property name: STRUCTURED-RESOURCE

Purpose: This property provides a typed reference to external information about a resource or optionally a plain text typed value.

Value type: There is no default value type for this property. The value type can be set to URI or TEXT.

Property Parameters: IANA, non-standard, label, restype or format type parameters can be specified on this property.

Conformance: This property MAY be specified zero or more times in any iCalendar component.

Description: When used in a component the value of this property provides information about resources used for the event.

When a LABEL parameter is supplied the language of the label must match that of the content and of the LANGUAGE parameter if present.

Format Definition:

This property is defined by the following notation:

```
  strucres       = "STRUCTURED-RESOURCE" strucresparam /
                   (
                     (
                       ";" "VALUE" "=" "URI"
                       ":" uri
                     )  /
                     (
                       ";" "VALUE" "=" "TEXT"
                       ":" text
                     )
                   )
                     CRLF

  strucresparam  = *(
                   ;
                   ; the following are OPTIONAL
                   ; but MUST NOT occur more than once
                   ;
                   (";" fmttypeparam) /
                   (";" labelparam) /
                   (";" languageparam) /
                   (";" restypeparam) /
                   ;
                   ; the following is OPTIONAL
                   ; and MAY occur more than once
                   ;
                   (";" other-param)
                   )
```

Example:

The following is an example of this property. It refers to a projector.

```
                  STRUCTURED-RESOURCE;restype="projector":
                    http://dir.example.com/projectors/3d.vcf
```

7.6. Structured-Data

Property Name: STRUCTURED-DATA

Purpose: This property specifies ancillary data associated with the calendar component.

Value Type: TEXT, BINARY or URI

Property Parameters: IANA, non-standard, inline encoding, and value data type property parameters can be specified on this property. The format type and schema parameters can be specified on this property and are RECOMMENDED for text or inline binary encoded content information.

Conformance: This property can be specified multiple times in an iCalendar object. Typically it would be used in "VEVENT", "VTODO", or "VJOURNAL" calendar components.

Description: This property is used to specify ancillary data in some structured format either directly (inline) as a "TEXT" or "BINARY" value, or as a link via a "URI" value.

Rather than define new iCalendar properties for the variety of event types that might occur, it would be better to leverage existing schemas for such data. For example, schemas available at https://schema.org include different event types. By using standard schemas, interoperability can be improved between calendar clients and non-calendaring systems that wish to generate or process the data.

This property allows the direct inclusion of ancillary data whose schema is defined elsewhere. This property also includes parameters to clearly identify the type of the schema being used so that clients can quickly and easily spot what is relevant within the calendar data and present that to users or process it within the calendaring system.

iCalendar does support an "ATTACH" property which can be used to include documents or links to documents within the calendar data. However, that property does not allow data to be included as a "TEXT" value (a feature that "STRUCTURED-DATA" does allow), plus attachments are often treated as "opaque" data to be processed by some other system rather than the calendar client. Thus the existing "ATTACH" property is not sufficient to cover the specific needs of inclusion of schema data. Extending the "ATTACH" property to support a new value type would likely cause interoperability problems. Thus a new property to support inclusion of schema data is warranted.

Format Definition:

This property is defined by the following notation:

```
  sdataprop   = "STRUCTURED-DATA" sdataparam
                      (":" text) /
                      (
                        ";" "ENCODING" "=" "BASE64"
                        ";" "VALUE" "=" "BINARY"
                        ":" binary
                      ) /
                      (
                        ";" "VALUE" "=" "URI"
                        ":" uri
                      )
                      CRLF

  sdataparam  = *(
                     ;
                     ; The following is OPTIONAL for a URI value,
                     ; RECOMMENDED for a TEXT or BINARY value,
                     ; and MUST NOT occur more than once.
                     ;
                     (";" fmttypeparam) /
                     (";" schemaparam) /
                     ;
                     ; The following is OPTIONAL,
                     ; and MAY occur more than once.
                     ;
                     (";" other-param)
                     ;
                )
```

Example: The following is an example of this property:

```
  STRUCTURED-DATA;FMTTYPE=application/ld+json;
   SCHEMA="https://schema.org/SportsEvent";
   VALUE=TEXT:{\n
     "@context": "http://schema.org"\,\n
     "@type": "SportsEvent"\,\n
     "homeTeam": "Pittsburgh Pirates"\,\n
     "awayTeam": "San Francisco Giants"\n
   }\n
```

8. New Components

8.1. Participant

Component name: PARTICIPANT

Purpose: This component provides information about a participant in an event or optionally a plain text typed value.

Conformance: This component MAY be appear in any iCalendar component.

Description: This component provides information about an participant in an event, task or poll. A participant may be an attendee in a scheduling sense and the ATTENDEE property may be specified in addition. Participants in events can be individuals or organizations, for example a soccer team, the spectators, or the musicians.

The SOURCE property if present may refer to an external definition of the participant - such as a vcard.

The STRUCTURED-ADDRESS property if present will provide a cal-address. If an ATTENDEE property has the same value the participant is considered schedulable. The PARTICIPANT component can be used to contain additional meta-data related to the attendee.

Format Definition:

This property is defined by the following notation:

```
    participantc  = "BEGIN" ":" "PARTICIPANT" CRLF
                    partprop *alarmc
                    "END" ":" "PARTICIPANT" CRLF

    partprop      = *(
                    ;
                    ; The following are REQUIRED,
                    ; but MUST NOT occur more than once.
                    ;
                    dtstamp  / participanttype /
                    ;
                    ; The following are OPTIONAL,
                    ; but MUST NOT occur more than once.
                    ;
                    created / description / last-mod / priority / seq /
                    source / status / scheduleaddress / summary / url /
                    ;
                    ; The following are OPTIONAL,
                    ; and MAY occur more than once.
                    ;
                    attach / categories / comment /
                    contact / rstatus / related /
                    resources / x-prop / iana-prop
                    ;
                    )
```

Note: When the PRIORITY is supplied it defines the ordering of PARTICIPANT components with the same value for the TYPE parameter.

Example:

The following is an example of this component. It contains a SOURCE property which points to a VCARD providing information about the event participant.

```
                    BEGIN:PARTICIPANT
                    PARTICIPANT-TYPE:PRINCIPAL_PERFORMER
                    SOURCE:http://dir.example.com/vcard/aviolinist.vcf
                    END:PARTICIPANT
```

Example:

The following is an example for the primary contact.

```
BEGIN: PARTICIPANT
SOURCE;FMTTYPE=text/vcard;
http://dir.example.com/vcard/contacts/contact1.vcf
PARTICIPANT-TYPE:PRIMARY-CONTACT
DESCRIPTION:A contact:
END:PARTICIPANT
```

8.2. Schedulable Participant

A PARTICIPANT component may represent someone or something that needs to be scheduled as defined for ATTENDEE in [RFC5545] and [RFC5546]. The PARTICIPANT component may also represent someone or something that is NOT to receive scheduling messages.

A PARTICIPANT component is defined to be schedulable if

- It contains a CALENDAR-ADDRESS property
- That property value is the same as the value for an ATTENDEE property.

If both of these conditions apply then the participant defined by the value of the URL property will take part in scheduling operations as defined in [RFC5546].

An appropriate use for the PARTICIPANT component in scheduling would be to store SEQUENCE and DTSTAMP properties associated with replies from each ATTENDEE. A LOCATION property within the PARTICIPANT component might allow better selection of meeting times when participants are in different timezones.

9. Participant Types

This section describes types of participation and provides registered values for the PARTICIPANT-TYPE property.

ACTIVE: A participant taking an active role - for example a team member.

INACTIVE: A participant taking an inactive part - for example an audience member.

SPONSOR: A sponsor of the event. The ORDER parameter may be used with this participant type to define the relative order of multiple sponsors.

CONTACT: Contact information for the event. The ORDER parameter may be used with this participant type to define the relative order of multiple contacts.

BOOKING-CONTACT: Contact information for reservations or payment

EMERGENCY-CONTACT: Contact in case of emergency

PUBLICITY-CONTACT: Contact for publicity

PLANNER-CONTACT: Contact for the event planner or organizer

PERFORMER: A performer - for example the soloist or the accompanist. The ORDER parameter may be used with this participant type to define the relative order of multiple performers. For example, ORDER=1 could define the principal performer or soloist.

SPEAKER: Speaker at an event

10. Resource Types

This section describes some initial resource types registered values for the RESTYPE parameter. Typically a resource is anything that might be required or used by a calendar entity and possibly has a directory entry.

Such resources may be a room or a projector. This registry provides a place in which such resources may be registered for use by scheduling sevices.

ROOM: A room for he event/meeting.

PROJECTOR: Projection equipment.

REMOTE-CONFERENCE-AUDIO: Audio remote conferencing facilities.

REMOTE-CONFERENCE-VIDEO: Video remote conferencing facilities.

11. Extended examples

The following are some examples of the use of the properties defined in this specification. They include additional properties defined in [RFC7986] which includes IMAGE.

11.1. Example 1

The following is an example of a VEVENT describing a concert. It includes location information for the venue itself as well as references to parking and restaurants.

```
BEGIN:VEVENT
CREATED:20170216T145739Z
DESCRIPTION: Piano Sonata No 3\n
 Piano Sonata No 30
DTSTAMP:20171116T145739Z
DTSTART;TZID=America/New_York:20170315T150000Z
DTEND;TZID=America/New_York:20170315T163000Z
LAST-MODIFIED:20170216T145739Z
SUMMARY:Beethoven Piano Sonatas
UID:123456
STRUCTURED-LOCATION;LABEL="The venue":
 http://dir.example.com/venues/big-hall.vcf
STRUCTURED-LOCATION;LABEL="The venue":
 http://dir.example.com/venues/parking.vcf
IMAGE;VALUE=URI;DISPLAY=BADGE;FMTTYPE=image/png:h
 ttp://example.com/images/concert.png
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:SPONSOR
SOURCE:http://example.com/sponsor.vcf
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:PERFORMER:
SOURCE:http://www.example.com/people/johndoe.vcf
END:PARTICIPANT
END:VEVENT
```

11.2. Example 2

The following is an example of a VEVENT describing a meeting. One of the attendees is a remote participant.

```
BEGIN:VEVENT
CREATED:20170216T145739Z
DTSTAMP:20101116T145739Z
DTSTART;TZID=America/New_York:20170315T150000Z
DTEND;TZID=America/New_York:20170315T163000Z
LAST-MODIFIED:20170216T145739Z
SUMMARY:Conference plaaning
UID:123456
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=ACCEPTED;CN=A:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CN=B:mailto:b@example.com
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:ACTIVE:
SOURCE:http://www.example.com/people/b.vcf
LOCATION:At home
END:PARTICIPANT
END:VEVENT
```

12. Security Considerations

Applications using these properties need to be aware of the risks entailed in using the URIs provided as values. See [RFC3986] for a discussion of the security considerations relating to URIs.

Security considerations relating to the "ATTACH" property, as described in [RFC5545], are applicable to the "STRUCTURED-DATA" property.

13. Privacy Considerations

Properties with a "URI" value type can expose their users to privacy leaks as any network access of the URI data can be tracked. Clients SHOULD NOT automatically download data referenced by the URI without explicit instruction from users. This specification does not introduce any additional privacy concerns beyond those described in [RFC5545].

14. IANA Considerations

This section defines updates to the tables defined in [RFC5545] and new tables.

14.1. Additional iCalendar Registrations

14.1.1. Property Registrations

This document defines the following new iCalendar properties to be added to the registry defined in Section 8.2.3 of [RFC5545]:

Property	Status	Reference
CALENDAR-ADDRESS	Current	RFCXXXX, Section 7.2
PARTICIPANT-TYPE	Current	RFCXXXX, Section 7.1
SOURCE	Current	RFCXXXX, Section 6
STRUCTURED-DATA	Current	RFCXXXX, Section 7.6
STYLED-DESCRIPTION	Current	RFCXXXX, Section 7.3
STRUCTURED-LOCATION	Current	RFCXXXX, Section 7.4
STRUCTURED-RESOURCE	Current	RFCXXXX, Section 7.5

14.1.2. Parameter Registrations

This document defines the following new iCalendar property parameters to be added to the registry defined in Section 8.2.4 of [RFC5545]:

Property Parameter	Status	Reference
LOCTYPE	Current	RFCXXXX, Section 5.1
ORDER	Current	RFCXXXX, Section 5.3
RESTYPE	Current	RFCXXXX, Section 5.2
SCHEMA	Current	RFCXXXX, Section 5.4

14.1.3. Component Registrations

This document defines the following new iCalendar components to be added to the registry defined in Section 8.3.1 of [RFC5545]:

Component	Status	Reference
PARTICIPANT	Current	RFCXXXX, Section 8.1

14.2. New Registration Tables

This section defines new registration tables for PARTICIPANT-TYPE and RESTYPE values. These tables maybe updated using the same approaches laid down in Section 8.2.1 of [RFC5545]

14.2.1. Participant Types Registry

The following table has been used to initialize the participant types registry.

Participant Type	Status	Reference
ACTIVE	Current	RFCXXXX, Section 9
INACTIVE	Current	RFCXXXX, Section 9
SPONSOR	Current	RFCXXXX, Section 9
CONTACT	Current	RFCXXXX, Section 9
BOOKING-CONTACT	Current	RFCXXXX, Section 9
EMERGENCY-CONTACT	Current	RFCXXXX, Section 9
PUBLICITY-CONTACT	Current	RFCXXXX, Section 9
PLANNER-CONTACT	Current	RFCXXXX, Section 9
PERFORMER	Current	RFCXXXX, Section 9
SPEAKER	Current	RFCXXXX, Section 9

14.2.2. Resource Types Registry

The following table has been used to initialize the resource types registry.

Resource Type	Status	Reference
PROJECTOR	Current	RFCXXXX, Section 10
ROOM	Current	RFCXXXX, Section 10
REMOTE-CONFERENCE-AUDIO	Current	RFCXXXX, Section 10
REMOTE-CONFERENCE-VIDEO	Current	RFCXXXX, Section 10

15. Acknowledgements

The author would like to thank Chuck Norris of eventful.com for his work which led to the development of this RFC.

The author would also like to thank the members of CalConnect, The Calendaring and Scheduling Consortium, the Event Publication

technical committee and the following individuals for contributing their ideas and support:

Cyrus Daboo, John Haug, Dan Mendell, Ken Murchison, Scott Otis,

16. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4589] Schulzrinne, H. and H. Tschofenig, "Location Types Registry", RFC 4589, DOI 10.17487/RFC4589, July 2006, <https://www.rfc-editor.org/info/rfc4589>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/info/rfc5545>.

[RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <https://www.rfc-editor.org/info/rfc5546>.

[RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <https://www.rfc-editor.org/info/rfc7986>.

[W3C.REC-xml-20060816] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium Recommendation REC-xml-20060816, August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>.

Appendix A. Open issues

None at the moment

Appendix B. Change log

calext-v04 2017-10-11 MD

- Change SCHEDULE-ADDRESS to CALENDAR-ADDRESS
- Explicitly broaden scope of SOURCE
- Add initial registry for RESTYPE and move new tables into separate section.
- Fix PARTTYPE/PARTICPANT-TYPE inconsistency

calext-v03 2017-10-09 MD

- Mostly typographical and other minor changes

calext-v02 2017-04-20 MD

- Add SCHEDULE-ADDRESS property
- PARTICIPANT becomes a component rather than a property. Turn many of the former parameters into properties.
- Use existing ATTENDEE property for scheduling.

calext-v01 2017-02-18 MD

- Change ASSOCIATE back to PARTICIPANT
- PARTICIPANT becomes a component rather than a property. Turn many of the former parameters into properties.

calext-v00 2016-08-?? MD

- Name changed - taken up by calext working group

v06 2016-06-26 MD

- Fix up abnf
- change ref to ietf from daboo
- take out label spec - use Cyrus spec

v05 2016-06-14 MD

- Remove GROUP and HASH. they can be dealt with elsewhere if desired

- Change ORDER to integer >= 1.
- Incorporate Structured-Data into this specification.

v04 2014-02-01 MD

- Added updates attribute.
- Minor typos.
- Resubmitted mostly to refresh the draft.

v03 2013-03-06 MD

- Replace PARTICIPANT with ASSOCIATE plus related changes.
- Added section showing modifications to components.
- Replace ID with GROUP and modify HASH.
- Replace TITLE param with LABEL.
- Fixed STYLED-DESCRIPTION in various ways, correct example.

v02 2012-11-02 MD

- Collapse sections with description of properties and the use cases into a section with sub-sections.
- New section to describe relating properties.
- Remove idref and upgrade hash to have the reference
- No default value types on properties..

v01 2012-10-18 MD Many changes.

- SPONSOR and STRUCTURED-CONTACT are now in PARTICIPANT
- Add a STRUCTURED-RESOURCE property
- STYLED-DESCRIPTION to handle rich text
- Much more...

2011-01-07

- Remove MEDIA - it’s going in the Cyrus RFC

- Rename EXTENDED-... to STRUCTURED-...

- Add TYPE parameter to SPONSOR

v00 2007-10-19 MD Initial version

Author's Address

Michael Douglass
Spherical Cow Group
226 3rd Street
Troy, NY 12180
USA

Email: mdouglass@sphericalcowgroup.com
URI: http://sphericalcowgroup.com

Network Working Group
Internet-Draft
Updates: 5545 (if approved)
Intended status: Standards Track
Expires: April 14, 2018

M. Douglass
Spherical Cow Group

October 11, 2017

Support for iCalendar Relationships
draft-ietf-calext-ical-relations-03

Abstract

This specification updates RELATED-TO defined in [RFC5545] and introduces new iCalendar properties LINK, CONCEPT and REFID to allow better linking and grouping of iCalendar components and related data.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 14, 2018.

Table of Contents

1. Introduction

iCalendar entities often need to be related to each other or to associated meta-data. These relationships can take the following forms

Structured iCalendar: iCalendar entities are related to each other in some structured way, for example as parent, sibling, before, after.

Grouped iCalendar: iCalendar entities are related to each other as a group. CATEGORIES are often used for this purpose but are problematic for application developers.

Linked: Entities are linked to each other through typed references.

1.1. Structured iCalendar relationships

The currently existing iCalendar [RFC5545] RELATED-TO property has no support for temporal relationships as used by standard project management tools.

The RELTYPE parameter is extended to take new values defining temporal relationships, a GAP parameter is defined to provide lead and lag values and RELATED-TO is extended to allow URI values. These changes allow the RELATED-TO property to define a richer set of relationships useful for project management.

1.2. Grouped iCalendar relationships

This specification defines a new REFID property which allows arbitrary groups of entities to be associated with the same key value.

REFID is used to identify a key allowing the association of tasks that are related to the same object and retrieval of a task based on this key. This may be, for example, to identify the tasks associated with a given project without having to communicate the task structure of the project, or, for example, in a package delivery system all tasks associated to a specific package.

As such, the presence of a REFID property imparts no meaning to the event. It is merely a key to allow retrieval. This is distinct from categorisation which, while allowing grouping also adds meaning to the entity to which it is attached.

1.3. Concept relationships

The name CONCEPT is used by the Simple Knowledge Organization System defined in [W3C.CR-skos-reference-20090317]. This more accurately defines what we mean by a catgeory. It’s not the words but the meaning.

The introduction of CONCEPT allows a more structured approach to categorization, with the possibility of namespaced and path-like values. Unlike REFID the CONCEPT property imparts some meaning. It is assumed that the value of this property will reference a well defined category.

The current [RFC5545] CATEGORY property is used as a free form ’tagging’ field. As such it is difficult to establish formal relationships between components based on their category.

Rather than attempt to add semantics to the current property it seeems best to continue its usage as an informal tag and establish a new property with more constraints.

1.4. Linked relationships

The currently existing iCalendar standard [RFC5545] lacks a general purpose method for referencing additional, external information relating to calendar components.

This document proposes a method for referencing typed external information that can provide additional information about an iCalendar component. This new LINK property is closely aligned to the LINK header defined in [RFC5988]

The LINK property defines a typed reference or relation to external meta-data or related resources. By providing type and format information as parameters, clients and servers are able to discover interesting references and make use of them, perhaps for indexing or the presentation of interesting links for the user.

It is often necessary to relate calendar components. The current RELATED-TO property only allows for a UID which is inadequate for many purposes. Allowing other types may help but might raise a number of backward compatibility issues. The link property can link components in different collections or even on different servers.

When publishing events it is useful to be able to refer back to the source of that information. The actual event may have been consumed from a feed or an ics file on a web site. A LINK property can provide a reference to the originator of the event.

Beyond the need to relate elements temporally, project management tools often need to be able to specify the relationships between the various events and tasks which make up a project. The LINK property provides such a mechanism.

The LINK property SHOULD NOT be treated as just another attachment. The ATTACH property is being extended to handle server-side management and stripping of inline data. Clients may choose to handle attachments differently as they are often an integral part of the message - for example, the agenda. See [I-D.daboo-caldav-attachments]

1.5. Caching and offline use

To facilitate offline display the link type may identify important pieces of data which should be downloaded in advance.

In general, the calendar entity should be self explanatory without the need to download referenced meta-data such as a web page.

1.6. Conventions Used in This Document

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

2. Reference Types

The actual reference value can take three forms specified by the type parameter

URI: The default type. This is a URI referring to the target.

UID: This allows for linking within a single collection and the value is assumed to be another component within that collection.

REFERENCE: An xpointer. In an XML environment it may be necessary to refer to an external XML artifact. The XPointer is defined in [W3C.WD-xptr-xpointer-20021219] and allows addressing portions of XML documents.

3. Link Relation Types

[RFC5988] defines two form of relation types, registered and extension. Registered relation types are added to a registry defined by [RFC5988] while extension relation types are specified as unique unregistered URIs, (at least unregistered in the [RFC5988] registry).

The relation types defined here will be registered with IANA in accordance with the specifications in [RFC5988].

4. Redefined Relation Type Value

Relationship parameter type values are defined in section 3.2.15. of [RFC5545]. This specification redefines that type to include the new temporal relationship values FINISHTOSTART, FINISHTOFINISH, STARTTOFINISH and STARTTOSTART. It also adds the DEPENDS-ON value to provide a link to a component upon which the current component depends.

Format Definition:

This property parameter is defined by the following notation:

```
  reltypeparam   = "RELTYPE" "="
                    ("PARENT"    ; Parent relationship - Default
                   / "CHILD"     ; Child relationship
                   / "SIBLING"   ; Sibling relationship
                   / "DEPENDS-ON" ; refers to previous task
                   / "REFID"     ; Relationship based on REFID
                   / "CONCEPT"
                          ; Relationship based on CONCEPT
                   / "FINISHTOSTART" ;  Temporal relationship
                   / "FINISHTOFINISH" ;  Temporal relationship
                   / "STARTTOFINISH" ;  Temporal relationship
                   / "STARTTOSTART" ;  Temporal relationship
                   / iana-token  ; Some other IANA-registered
                                 ; iCalendar relationship type
                   / x-name)     ; A non-standard, experimental
                                 ; relationship type
```

Description: This parameter can be specified on a property that references another related calendar component. The parameter may specify the hierarchical relationship type of the calendar component referenced by the property when the value is PARENT, CHILD or SIBLING. If this parameter is not specified on an allowable property, the default relationship type is PARENT. Applications MUST treat x-name and iana-token values they don't recognize the same way as they would the PARENT value.

This parameter defines the temporal relationship when the value is one of the project management standard relationships FINISHTOSTART, FINISHTOFINISH, STARTTOFINISH or STARTTOSTART. This property will be present in the predecessor entity and will refer to the successor entity. The GAP parameter specifies the lead or lag time between the predecessor and the successor. In the description of each temporal relationship below we refer to Task-A which contains and controls the relationship and Task-B the target of the relationship.

RELTYPE=PARENT: See [RFC5545] section 3.2.15.

RELTYPE=CHILD: See [RFC5545] section 3.2.15.

RELTYPE=SIBLING: See [RFC5545] section 3.2.15.

RELTYPE=DEPENDS-ON: Indicates that the current calendar component depends on the referenced calendar component in some manner. For

example a task may be blocked waiting on the other, referenced, task.

RELTYPE=REFID: Establishes a reference from the current component to components with a REFID property which matches the value given in the associated RELATED-TO property.

RELTYPE=CONCEPT: Establishes a reference from the current component to components with a CONCEPT property which matches the value given in the associated RELATED-TO property.

RELTYPE=FINISHTOSTART: Task-B cannot start until Task-A finishes. For example, when sanding is complete, painting can begin.

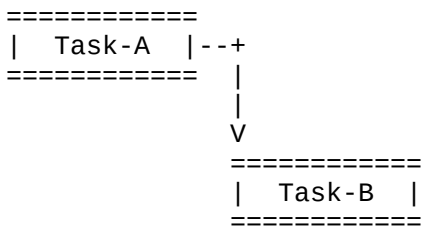

Figure 1: Finish to start relationship

RELTYPE=FINISHTOFINISH: Task-B cannot finish before Task-A is finished, that is the end of Task-A defines the end of Task-B. For example, we start the potatoes, then the meat then the peas but they should all be cooked at the same time.

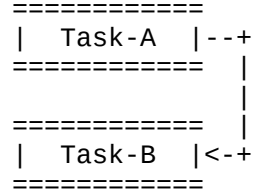

Figure 2: Finish to finish relationship

RELTYPE=STARTTOFINISH: The start of Task-A (which occurs after Task-B) controls the finish of Task-B. For example, ticket sales (Task-B) end when the game starts (Task-A).

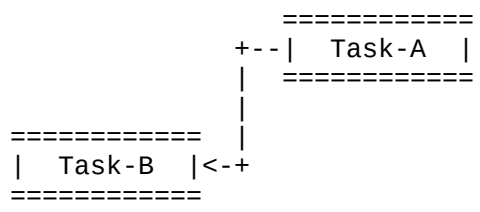

Figure 3: Start to finish relationship

RELTYPE=STARTTOSTART: The start of Task-A triggers the start of Task-B, that is Task-B can start anytime after Task-A starts.

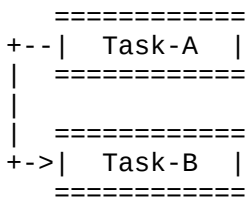

Figure 4: Start to start relationship

5. New Property Parameters

5.1. Rel

Parameter name: REL

Purpose: To specify the relationship of data referenced by a LINK property.

Format Definition:

This parameter is defined by the following notation:

```
relparam     = "REL" "="
               ("SOURCE"      ; Link to source of this component
               / DQUOTE uri DQUOTE
               / x-name        ; Experimental reference type
               / iana-token)   ; Other IANA registered type
```

Description: This parameter MUST be specified on all LINK properties, and defines the type of reference. This allows programs consuming this data to automatically scan for references they support. In addition to the values defined here any value

defined in [RFC5988] may be used. There is no default relation type.

REL=SOURCE: identifies the source of the event information.

Registration: These relation types are registered in [RFC5988]

5.2. Gap

Parameter name: GAP

Purpose: To specify the length of the gap, positive or negative between two temporaly related components.

Format Definition:

This parameter is defined by the following notation:

```
gapparam      = "GAP" "=" dur-value
```

Description: This parameter MAY be specified on the RELATED-TO property, and defines the duration of time between the predecessor and successor in an interval. When positive it defines the lag time between a task and its logical successor. When negative it defines the lead time.

An example of lag time might be if task A is "paint the room" and task B is "hang the drapes" then task A may be related to task B with RELTYPE=FINISHTOSTART with a gap long enough for the paint to dry.

An example of lead time might be to relate a 1 week task A to the end of task B with RELTYPE=STARTTOFINISH and a negative gap of 1 week so they finish at the same time.

6. New Value Data Types

This specification defines the following new value types to be used with the VALUE property parameter:

UID VALUE=UID indicates that the associated value is the UID for a component.

REFERENCE VALUE=REFERENCE indicates that the associated value is an xpointer referencing an associated XML artifact.

7. New Properties

7.1. Concept

Property name: CONCEPT

Purpose: This property defines the formal categories for a calendar component.

Value type: URI

Property Parameters: IANA, and non-standard parameters can be specified on this property.

Conformance: This property can be specified zero or more times in any iCalendar component.

Description: This property is used to specify formal categories or classifications of the calendar component. The values are useful in searching for a calendar component of a particular type and category.

Within the "VEVENT", "VTODO", or "VJOURNAL" calendar components, more than one formal category can be specified by using multiple CONCEPT properties.

This categorization is distinct from the more informal "tagging" of components provided by the existing CATEGORIES property. It is expected that the value of the CONCEPT property will reference an external resource which provides information about the categorization.

In addition, a structured URI value allows for hierarchical categorization of events.

Possible category resources are the various proprietary systems, for example Library of Congress, or an open source derived from something like the dmoz.org data.

Format Definition:

This property is defined by the following notation:

```
  concept        =  "CONCEPT" conceptparam ":"
                        uri CRLF

  conceptparam = *(
                ;
                ; The following is OPTIONAL,
                ; and MAY occur more than once.
                ;
                (";" other-param)
                ;
                )
```

Example:

The following is an example of this property. It points to a server acting as the source for the calendar object.

```
  SCONCEPT:http://example.com/event-types/sports
  CONCEPT:http://example.com/event-types/arts/music
  CONCEPT:http://example.com/task-types/delivery
```

7.2. Link

Property name: LINK

Purpose: This property provides a reference to external information about a component.

Value type: URI, TEXT or REFERENCE

Property Parameters: Non-standard, reference type or format type parameters can be specified on this property.

Conformance: This property MAY be specified in any iCalendar component.

Description: When used in a component the value of this property points to additional information related to the component. For example, it may reference the originating web server.

Format Definition:

This property is defined by the following notation:

```
  link            = "LINK" linkparam  /
                    (
                      ";" "VALUE" "=" "TEXT"
                      ":" text
                    )
                    (
                      ";" "VALUE" "=" "REFERENCE"
                      ":" text
                    )
                    (
                      ";" "VALUE" "=" "URI"
                      ":" uri
                    )
                    CRLF


  linkparam       = *(

                  ; the following is MANDATORY
                  ; and MAY occur more than once

                  (";" relparam) /

                  ; the following are MANDATORY
                  ; but MUST NOT occur more than once

                  (";" fmttypeparam) /
                  (";" labelparam) /
                  ; labelparam is defined in ...

                  ; the following is OPTIONAL
                  ; and MAY occur more than once

                  (";" xparam)

                  )
```

Example:

The following is an example of this property. It points to a server acting as the source for the calendar object.

```
  LINK;REL=SOURCE;LABEL=The Egg:http://example.com/events
```

7.3. Refid

Property name: REFID

Purpose: This property value acts as a key for associated iCalendar entities.

Value type: TEXT

Property Parameters: Non-standard parameters can be specified on this property.

Conformance: This property MAY be specified multiple times in any iCalendar component.

Description: The value of this property is a text identifier that allows associated components to be located or retrieved as a whole. For example all of the events in a travel itinerary would have the same REFID value.

Format Definition:

This property is defined by the following notation:

```
refid      = "REFID" refidparam ":" text CRLF


refidparam      = *(

                ; the following is OPTIONAL
                ; and MAY occur more than once

                (";" xparam)

                )
```

Example:

The following is an example of this property.

```
REFID:itinerary-2014-11-17
```

8. Redefined RELATED-TO Property

8.1. RELATED-TO

Property name: RELATED-TO

Purpose: This property is used to represent a relationship or reference between one calendar component and another. The definition here extends the definition in Section 3.8.4.5. of [RFC5545] by allowing URI or UID values and a GAP parameter.

Value type: URI, UID or TEXT

Property Parameters: Relationship type, IANA and non-standard property parameters can be specified on this property.

Conformance: This property MAY be specified in any iCalendar component.

Description: By default or when VALUE=UID is specified, the property value consists of the persistent, globally unique identifier of another calendar component. This value would be represented in a calendar component by the "UID" property.

By default, the property value points to another calendar component that has a PARENT relationship to the referencing object. The "RELTYPE" property parameter is used to either explicitly state the default PARENT relationship type to the referenced calendar component or to override the default PARENT relationship type and specify either a CHILD or SIBLING relationship or a temporal relationship.

The PARENT relationship indicates that the calendar component is a subordinate of the referenced calendar component. The CHILD relationship indicates that the calendar component is a superior of the referenced calendar component. The SIBLING relationship indicates that the calendar component is a peer of the referenced calendar component.

The FINISHTOSTART, FINISHTOFINISH, STARTTOFINISH or STARTTOSTART relationships define temporal relationships as specified in the reltype parameter definition.

Changes to a calendar component referenced by this property can have an implicit impact on the related calendar component. For example, if a group event changes its start or end date or time, then the related, dependent events will need to have their start and end dates changed in a corresponding way. Similarly, if a PARENT calendar component is cancelled or deleted, then there is an implied impact to the related CHILD calendar components. This

property is intended only to provide information on the relationship of calendar components. It is up to the target calendar system to maintain any property implications of this relationship.

Format Definition:

This property is defined by the following notation:

```
related    = "RELATED-TO" relparam ( ":" text ) /
             (
               ";" "VALUE" "=" "UID"
               ":" uid
             )
             (
               ";" "VALUE" "=" "URI"
               ":" uri
             )
             CRLF

relparam   = *(
            ;
            ; The following are OPTIONAL,
            ; but MUST NOT occur more than once.
            ;
            (";" reltypeparam) /
            (";" gapparam) /
            ;
            ; The following is OPTIONAL,
            ; and MAY occur more than once.
            ;
            (";" other-param)
            ;
            )
```

Example:

The following are examples of this property.

```
RELATED-TO:jsmith.part7.19960817T083000.xyzMail@example.com

RELATED-TO:19960401-080045-4000F192713-0052@example.com

RELATED-TO;VALUE=URI;RELTYPE=STARTTOFINISH:
 http://example.com/caldav/user/jb/cal/
 19960401-080045-4000F192713.ics
```

9. Security Considerations

Applications using the LINK property need to be aware of the risks entailed in using the URIs provided as values. See [RFC3986] for a discussion of the security considerations relating to URIs.

The CONCEPT and redefined RELATED-TO property have the same issues in that values may be URIs.

10. IANA Considerations

10.1. iCalendar Property Registrations

The following iCalendar property names have been added to the iCalendar Properties Registry defined in Section 8.3.2 of [RFC5545]

Property	Status	Reference
CONCEPT	Current	Section 7.1
LINK	Current	Section 7.2
REFID	Current	Section 7.3

10.2. iCalendar Property Parameter Registrations

The following iCalendar property parameter names have been added to the iCalendar Parameters Registry defined in Section 8.3.3 of [RFC5545]

Parameter	Status	Reference
REL	Current	Section 5.1
GAP	Current	Section 5.2

10.3. iCalendar Value Data Type Registrations

The following iCalendar property parameter names have been added to the iCalendar Value Data Types Registry defined in Section 8.3.4 of [RFC5545]

Value Data Type	Status	Reference
UID	Current	Section 6
REFERENCE	Current	Section 6

10.4. iCalendar RELTYPE Value Registrations

The following iCalendar "RELTYPE" values have been added to the iCalendar Relationship Types Registry defined in Section 8.3.8 of [RFC5545]

Relationship Type	Status	Reference
DEPENDS-ON	Current	Section 4
REFID	Current	Section 4
CONCEPT	Current	Section 4
FINISHTOSTART	Current	Section 4
FINISHTOFINISH	Current	Section 4
STARTTOFINISH	Current	Section 4
STARTTOSTART	Current	Section 4

10.5. New Reference Type Registration

The following link relation values have been added to the Reference Types Registry defined in Section 6.2.2 of [RFC5988]

Name	Status	Reference
SOURCE	Current	Section 5.1

11. Acknowledgements

The author would like to thank the members of the Calendaring and Scheduling Consortium technical committees and the following individuals for contributing their ideas, support and comments:

Adrian Apthorp, Cyrus Daboo, Marten Gajda, Ken Murchison

The author would also like to thank the Calendaring and Scheduling Consortium for advice with this specification.

12. Normative References

[I-D.daboo-caldav-attachments]
Daboo, C. and A. Quillaud, "CalDAV Managed Attachments", draft-daboo-caldav-attachments-03 (work in progress), February 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/info/rfc5545>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <https://www.rfc-editor.org/info/rfc5988>.

[W3C.CR-skos-reference-20090317]
Bechhofer, S. and A. Miles, "SKOS Simple Knowledge Organization System Reference", World Wide Web Consortium CR CR-skos-reference-20090317, March 2009, <http://www.w3.org/TR/2009/CR-skos-reference-20090317>.

[W3C.REC-xml-20060816]
Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium Recommendation REC-xml-20060816, August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>.

[W3C.WD-xptr-xpointer-20021219]
DeRose, S., Daniel, R., and E. Maler, "XPointer xpointer() Scheme", World Wide Web Consortium WD WD-xptr-xpointer-20021219, December 2002, <http://www.w3.org/TR/2002/WD-xptr-xpointer-20021219>.

Author's Address

Michael Douglass
Spherical Cow Group
226 3rd Street
Troy, NY 12180
USA

Email: mdouglass@sphericalcowgroup.com
URI: http://sphericalcowgroup.com

Calendaring extensions
Internet-Draft
Intended status: Standards Track
Expires: August 2, 2018

N. Jenkins
R. Stepanek
FastMail

January 29, 2018

JSCalendar: A JSON representation of calendar data
draft-ietf-calext-jscalendar-00

Abstract

This specification defines a data model and JSON representation of calendar data that can be used for storage and data exchange in a calendaring and scheduling environment. It aims to be an alternative to the widely deployed iCalendar data format and to be unambiguous, extendable and simple to process.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 2, 2018.

Table of Contents

1. Introduction

This document defines a data model for calendar event and task objects, or groups of such objects, in electronic calendar applications and systems. It aims to be unambiguous, extendable and simple to process.

The key design considerations for this data model are as follows:

- The attributes of the calendar entry represented must be described as a simple key-value pair, reducing complexity of its representation.
- The data model should avoid all ambiguities and make it difficult to make mistakes during implementation.
- Most of the initial set of attributes should be taken from the iCalendar data format ([RFC5545], also see Section 1.1), but the specification should add new attributes or value types, or not support existing ones, where appropriate. Conversion between the data formats need not fully preserve semantic meaning.
- Extensions, such as new properties and components, MUST NOT lead to requiring an update to this document.

The representation of this data model is defined in the I-JSON format [RFC7493], which is a strict subset of the JavaScript Object Notation (JSON) Data Interchange Format [RFC8259]. Using JSON mostly is a pragmatic choice: its widespread use should help to speed up JSCalendar adoption and a wide range of production-ready JSON implementations allows to decrease interoperability issues.

1.1. Relation to the iCalendar format

The iCalendar data format [RFC5545], a widely deployed interchange format for calendaring and scheduling data, has served calendaring vendors for a long while, but contains some ambiguities and pitfalls that can not be overcome without backward-incompatible changes.

For example, iCalendar defines various formats for local times, UTC time and dates, which confuses new users. Other sources for errors are the requirement for custom time-zone definitions within a single calendar component, as well as the iCalendar format itself; the latter causing interoperability issues due to misuse of CR LF terminated strings, line continuations and subtle differences between iCalendar parsers. Lastly, up until recently the iCalendar format

did not allow to express the difference between two calendar components, which results in verbose exchanges during scheduling.

Some of these issues were addressed by the jCal [RFC7265] format, which is a direct mapping between iCalendar and JSON. However, it did not attempt to extend or update iCalendar semantics.

1.2. Notational Conventions

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].

The underlying format used for this specification is JSON. Consequently, the terms "object" and "array" as well as the four primitive types (strings, numbers, booleans, and null) are to be interpreted as described in Section 1 of[RFC8259].

Some examples in this document contain "partial" JSON documents used for illustrative purposes. In these examples, three periods "..." are used to indicate a portion of the document that has been removed for compactness.

2. JSCalendar objects

This section describes the calendar object types specified by JSCalendar.

2.1. JSEvent

MIME type: "application/calendar+json;type=jsevent"

A JSEvent represents a scheduled amount of time on a calendar, typically a meeting, appointment, reminder or anniversary. Multiple participants may partake in the event at multiple locations.

The @type (Section 4.1.1) property value MUST be "jsevent".

2.2. JSTask

MIME type: "application/calendar+json;type=jstask"

A JSTask represents an action-item, assignment, to-do or work item .

The @type (Section 4.1.1) property value MUST be "jstask".

A JSTask may start and be due at certain points in time, may take some estimated time to complete and may recur; none of which is

required. This notably differs from JSEvent (Section 2.1) which is required to start at a certain point in time and typically takes some non-zero duration to complete.

2.3. JSGroup

MIME type: "application/calendar+json;type=jsgroup"

A JSGroup is a collection of JSEvent (Section 2.1) and JSTask (Section 2.2) objects. Typically, objects are grouped by topic (e.g. by keywords) or calendar membership.

The @type (Section 4.1.1) property value MUST be "jsgroup".

3. Structure of JSCalendar objects

A JSCalendar object is a JSON object, which MUST be valid I-JSON (a stricter subset of JSON), as specified in [RFC8259]. Property names and values are case-sensitive.

The object has a collection of properties, as specified in the following sections. Unless otherwise specified, all properties are optional; omitted properties MUST be treated identically to if that property had the value of "null", unless otherwise specified.

3.1. Type signatures

Types signatures are given for all JSON objects in this document. The following conventions are used:

- "Boolean|String": The value is either a JSON "Boolean" value, or a JSON "String" value.
- "Foo": Any name that is not a native JSON type means an object for which the properties (and their types) are defined elsewhere within this document.
- "Foo[]": An array of objects of type "Foo".
- "String[Foo]": A JSON "Object" being used as a map (associative array), where all the values are of type "Foo".

3.2. Data Types

In addition to the standard JSON data types, the following data types are used in this specification:

3.2.1. UTCDate

This is a string in [RFC3339] "date-time" format, with the further restrictions that any letters MUST be in upper-case, the time component MUST be included and the time MUST be in UTC. Fractional second values MUST NOT be included unless non-zero and MUST NOT have trailing zeros, to ensure there is only a single representation for each date-time.

For example "2010-10-10T10:10:10.003Z" is OK, but "2010-10-10T10:10:10.000Z" is invalid and MUST be encoded as "2010-10-10T10:10:10Z".

In common notation, it should be of the form "YYYY-MM-DDTHH:MM:SSZ".

3.2.2. LocalDate

This is a date-time string _with no time-zone/offset information_. It is otherwise in the same format as UTCDate: "YYYY-MM-DDTHH:MM:SS". The time-zone to associate the LocalDate with comes from an associated property, or if no time-zone is associated it defines _floating time_. Floating date-times are not tied to any specific time-zone. Instead, they occur in every timezone at the same _wall-clock_ time (as opposed to the same instant point in time).

3.2.3. Duration

A duration is represented by a subset of ISO8601 duration format, as specified by the following ABNF:

```
dur-secfrac = "." 1*DIGIT
dur-second  = 1*DIGIT [dur-secfrac] "S"
dur-minute  = 1*DIGIT "M" [dur-second]
dur-hour    = 1*DIGIT "H" [dur-minute]
dur-time    = "T" (dur-hour / dur-minute / dur-second)
dur-day     = 1*DIGIT "D"

duration    = "P" (dur-day [dur-time] / dur-time)
```

In addition, the duration MUST NOT include fractional second values unless the fraction is non-zero. A zero duration MUST be represented as "P0D".

3.2.4. PatchObject

A *PatchObject* is of type "String[*|null]", and represents an unordered set of patches on a JSON object. The keys are a path in a subset of [RFC6901] JSON pointer format, with an implicit leading "/"

(i.e. prefix each key with "/" before applying the JSON pointer evaluation algorithm).

A patch within a PatchObject is only valid, if all of the following conditions apply:

1. The pointer MUST NOT reference inside an array (i.e. it MUST NOT insert/delete from an array; the array MUST be replaced in its entirety instead).

2. When evaluating a path, all parts prior to the last (i.e. the value after the final slash) MUST exist.

3. There MUST NOT be two patches in the PatchObject where the pointer of one is the prefix of the pointer of the other, e.g. "alerts/1/offset" and "alerts".

The value associated with each pointer is either:

- "null": Remove the property from the patched object. If not present in the parent, this a no-op.

- Anything else: The value to replace the inherited property on the patch object with (if present) or add to the property (if not present).

Implementations MUST reject a PatchObject if any of its patches are invalid.

3.2.5. Normalisation and equivalence

JSCalendar aims to provide unambiguous definitions for value types and properties, but does not define a general normalisation or equivalence method for JSCalendar objects and types. This is because the notion of equivalence might range from byte-level equivalence to semantic equivalence, depending on the respective use case (for example, the CalDAV protocol [RFC4791] requires octet equivalence of the encoded calendar object to determine ETag equivalence).

Normalisation of JSCalendar objects is hindered because of the following reasons:

- Custom JSCalendar properties may contain arbitrary JSON values, including arrays. However, equivalence of arrays might or might not depend on the order of elements, depending on the respective property definition.

- Several JSCalendar property values are defined as URIs and MIME types, but normalisation of these types is inherently protocol and scheme-specific, depending on the use-case of the equivalence definition (see section 6 of [RFC3986]).

Considering this, the definition of equivalence and normalisation is left to client and server implementations and to be negotiated by a calendar exchange protocol or defined by another RFC.

3.3. Custom property extensions and values

Vendors MAY add additional properties to the calendar object to support their custom features. The names of these properties MUST be prefixed with a domain name controlled by the vendor to avoid conflict, e.g. "example.com/customprop".

Some JSCalendar properties allow vendor-specific value extensions. If so, vendor specific values MUST be prefixed with a domain name controlled by the vendor, e.g. "example.com/customrel", unless otherwise noted.

4. Common JSCalendar properties

This section describes the properties that are common to the various JSCalendar object types. Specific JSCalendar object types may only support a subset of these properties. The object type definitions in section Section 5 describe the set of supported properties per type.

4.1. Metadata properties

4.1.1. @type

Type: "String"

Specifies the type which this object represents. This MUST be one of the following values, registered in a future RFC, or a vendor-specific value:

- "jsevent": a JSCalendar event (Section 2.1).
- "jstask": a JSCalendar task (Section 2.2).
- "jsgroup": a JSCalendar group (Section 2.3).

A valid JSCalendar object MUST include this property.

4.1.2. uid

Type: "String"

A globally unique identifier, used to associate the object as the same across different systems, calendars and views. The value of this property MUST be unique across _all_ JSCalendar objects, even if they are of different type. [RFC4122] describes a range of established algorithms to generate universally unique identifiers (UUID), and the random or pseudo-random version is recommended to use.

A valid JSCalendar object MUST include this property.

4.1.3. relatedTo

Type: "String[Relation]|null"

Relates the object to other JSCalendar objects. This is represented as a map of the uids of the related objects to information about the relation.

A *Relation* object has the following properties:

- *relation*: "String[]" Describes how the linked object is related to this object.

  The strings in the array MUST each be at most one of the following values, registered in a future RFC, or a vendor-specific value:

  - "first": The linked object is the first in the series this object is part of.
  - "next": The linked object is the next in the series this object is part of.
  - "child": The linked object is a subpart of this object.
  - "parent": This object is part of the overall linked object.

If an object is split to make a "this and future" change to a recurrence, the original object MUST be truncated to end at the previous occurrence before this split, and a new object created to represent all the objects after the split. A "relation=["next"]" relatedTo property MUST be set on the original object with the uid of the new object. A "relation=["first"]" relatedTo property with the UID of the first object in the series MUST be set on the new object.

Clients can then follow these UIDs to get the complete set of objects if the user wishes to modify them all at once.

4.1.4. prodId

Type: "String|null"

The identifier for the product that created the JSCalendar object.

The vendor of the implementation SHOULD ensure that this is a globally unique identifier, using some technique such as an FPI value, as defined in [ISO.9070.1991].

This property SHOULD NOT be used to alter the interpretation of an JSCalendar object beyond the semantics specified in this document. For example, it is not to be used to further the understanding of non-standard properties.

4.1.5. created

Type: "UTCDate|null"

The date and time this object was initially created.

4.1.6. updated

Type: "UTCDate"

The date and time the data in this object was last modified.

4.1.7. sequence

Type: "Number" (Defaults to "0" if omitted)

Initially zero, this MUST be a non-negative integer that is monotonically incremented each time a change is made to the object.

4.1.8. method

Type: "String|null"

The iTIP ([RFC5546]) method, in lower-case. Used for scheduling.

4.2. What and where properties

4.2.1. title

Type: "String" (Defaults to the empty string if omitted)

A short summary of the object.

4.2.2. description

Type: "String" (Defaults to the empty string if omitted)

A longer form description of the object. This is plain text, but a client SHOULD attempt to hyperlink URLs when displaying it.

4.2.3. htmlDescription

Type: "String" (Defaults to the empty string if omitted)

A longer form rich-text description of the object. This is HTML text [1] and allows to reference resources in the *links* property by use of CID URLs (see [RFC2392]). To convert a CID URL to the *cid* property value of a *Link* object, implementations MUST follow the conversion described in section 2 of [RFC2392]. Implementations MAY choose not to follow untrusted external CID URLs referenced in the *links* property, in which case they MUST treat the *htmlDescription* property as if omitted. Implementations MUST preserve the value of this property, even if it contains untrusted links.

4.2.4. locations

Type: "String[Location]|null"

A map of location ids to Location objects, representing locations associated with the object. A location id may be any string and need only be unique to this object, although a UUID is a practical choice.

A *Location* object has the following properties. All properties are optional, but every Location object MUST have at least one property:

- *name*: "String" The human-readable name of the location.
- *description*: "String" Human-readable instructions for accessing this location. This may be an address, set of directions, door access code, etc.
- *rel*: "String" The relation type of this location to the JSCalendar object.

This MUST be either one of the following values, registered in a future RFC, or a vendor-specific value. Any value the client or server doesn't understand should be treated the same as "unknown".

* "start": The JSCalendar object starts at this location.

* "end": The JSCalendar object ends at this location.

* "virtual": This is not a physical location (e.g. this location is an online chat room where people will meet).

* "unknown": The relation of this location to the calendar object is unknown.

- *features*: "String[]|null" The features supported by this location.

  The strings in the array MUST each be either one of the following values, registered in a future RFC, or a vendor-specific value. Any value the client or server doesn't understand should be ignored, but preserved.

  The features supported by locations with rel-type "virtual" are:

  * "audio": audio conferencing

  * "chat": chat or instant messaging

  * "screen": screen sharing

  * "video": video conferencing

  * any vendor-prefixed custom value

- *timeZone*: "String|null" (Defaults to "null" if omitted) A time-zone for this location.

  If "null", the *timeZone* from the JSCalendar object MUST be presumed when a time-zone is needed in relation to this location.

- *coordinates*: "String" An [RFC5870] "geo:" URI for the location.

- *uri*: "String" A URI that represents how to connect to this location.

  This may be a telephone number (represented as "tel:+1-555-555-555") for a teleconference, a web address for online chat, or any custom URI.

- *linkIds*: "String[]|null" A list of ids for links to alternate representations of this location.

  For example, an alternative representation could be in vCard format. If a given value does not correspond to any link id in the links property of the instance, this MUST be ignored.

4.2.5. links

Type: "String[Link]|null"

A map of link ids to Link objects, representing external resources associated with the object. A link id may be any string and need only be unique to this object, although the href or a UUID are practical choices.

A *Link* object has the following properties:

- *href*: "String" A URI from which the resource may be fetched.

  This MAY be a "data:" URL, but it is recommended that the file be hosted on a server to avoid embedding arbitrary large data in JSCalendar object instances.

- *cid* "String|null" The id used within the *htmlDescription* property to reference this link.

  If not null, this MUST be a valid Content-ID MIME header value (see [RFC2392]). The identifier MUST be unique within this JSCalendar object but has no meaning beyond that. Specifically, it MAY be different from the *Link* object identifier in the enclosing *links* property.

- *type*: "String|null"(optional, defaults to "null") The content-type [RFC6838] of the resource, if known.

- *size*: "Number|null"(optional, defaults to "null") The size, in bytes, of the resource when fully decoded (i.e. the number of bytes in the file the user would download), if known.

- *rel*: "String"(optional, defaults to "related") Identifies the relation of the linked resource to the object. The value MUST be a registered relation type (see [RFC8288] and IANA Link Relations [2]).

  Links with a rel of "enclosure" SHOULD be considered by the client as attachments for download.

Links with a rel of "describedby" SHOULD be considered by the client to be an alternate representation of the description and HTML description.

Links with a rel of "icon" SHOULD be considered by the client to be an image that it MAY use when presenting the calendar data to a user. The *properties* object of this link MAY include a "display" property indicating the intended purpose of this image. If included, the value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value.

* "badge": an image inline with the title of the object

* "graphic": a full image replacement for the object itself

* "fullsize": an image that is used to enhance the object

* "thumbnail": a smaller variant of "fullsize " to be used when space for the image is constrained

- *title*: "String|null"(optional, defaults to "null") A human-readable description of the resource.

- *properties*: "String[String|null]|null"(optional, defaults to "null") Extra metadata submitted by clients about a link. Server implementations MUST preserve these properties.

  The keys are as defined in this document, as defined in a future RFC, or URIs that should be owned by the client author to avoid conflicts.

4.2.6. locale

Type: "String|null"

The [RFC5646] language tag that best describes the locale used for the calendar object, if known.

4.2.7. keywords

Type: "String[]|null"

A list of keywords or tags related to the object. The values are free-form and do not have to follow any particular structure.

4.2.8. categories

Type: "String[]|null"

Specifies the categories related to the calendar object. Array values MUST be URIs. In contrast to *keywords*, categories typically are structured.

For example, a vendor owning the domain "example.com" might define the categories "http://example.com/categories/sports/american-football"" and "http://example.com/categories/music/r-b".

4.2.9. color

Type: "String|null"

Specifies a color clients MAY use when displaying this calendar object. The value is a case-insensitive color name taken from the CSS3 set of names, defined in Section 4.3 of W3C.REC-css3-color-20110607 [3] or a CSS3 RGB color hex value.

4.3. Recurrence properties

4.3.1. recurrenceRule

Type: "Recurrence"

Defines a recurrence rule (repeating pattern) for recurring calendar objects.

A *Recurrence* object is a JSON object mapping of a RECUR value type in iCalendar, see [RFC5545] and[RFC7529]. Objects recur by applying the recurrence rule (and *recurrenceOverrides*) to the *start* date/time. A JSTask (Section 2.2) without a *start* property value recurs by its *due* date/time, if defined.

A Recurrence object has the following properties:

- *frequency*: "String" This MUST be one of the following values:
  - "yearly"
  - "monthly"
  - "weekly"
  - "daily"

  * "hourly"

  * "minutely"

  * "secondly"

  To convert from iCalendar, simply lower-case the FREQ part.

- *interval*: "Number"(optional, defaults to "1") The INTERVAL part from iCal. If included, it MUST be an integer "x >= 1".

- *rscale*: "String"(optional, defaults to ""gregorian"") The RSCALE part from iCalendar RSCALE [RFC7529], converted to lower-case.

- *skip*: "String"(optional, defaults to ""omit"") The SKIP part from iCalendar RSCALE [RFC7529], converted to lower-case.

- *firstDayOfWeek*: "String"(optional, defaults to ""mo"") The WKST part from iCalendar, represented as a lower-case abbreviated two-letter English day of the week. If included, it MUST be one of the following values: ""mo"|"tu"|"we"|"th"|"fr"|"sa"|"su"".

- *byDay*: "NDay[]"(optional) An *NDay* object has the following properties:

  * *day*: "String" The day-of-the-week part of the BYDAY value in iCalendar, lower-cased. MUST be one of the following values: ""mo"|"tu"|"we"|"th"|"fr"|"sa"|"su"".

  * *nthOfPeriod*: "Number"(optional) The optional ordinal part of the BYDAY value in iCalendar (e.g. ""+1"" or ""-3""). If present, rather than representing every occurrence of the weekday defined in the *day* property of this *NDay*, it represents only a specific instance within the recurrence period. The value can be positive or negative, but MUST NOT be zero. A negative integer means nth-last of period.

- *byMonthDay*: "Number[]"(optional) The BYMONTHDAY part from iCalendar. The array MUST have at least one entry if included.

- *byMonth*: "String[]"(optional) The BYMONTH part from iCalendar. Each entry is a string representation of a number, starting from "1" for the first month in the calendar (e.g. ""1" " means ""January"" with Gregorian calendar), with an optional ""L"" suffix (see [RFC7529]) for leap months (this MUST be upper-case, e.g. ""3L""). The array MUST have at least one entry if included.

- *byYearDay*: "Number[]"(optional) The BYYEARDAY part from iCalendar. The array MUST have at least one entry if included.

- *byWeekNo*: "Number[]"(optional) The BYWEEKNO part from iCalendar. The array MUST have at least one entry if included.

- *byHour*: "Number[]"(optional) The BYHOUR part from iCalendar. The array MUST have at least one entry if included.

- *byMinute*: "Number[]"(optional) The BYMINUTE part from iCalendar. The array MUST have at least one entry if included.

- *bySecond*: "Number[]"(optional) The BYSECOND part from iCalendar. The array MUST have at least one entry if included.

- *count*: "Number"(optional) The COUNT part from iCalendar. This MUST NOT be included if an *until* property is specified.

- *until*: "LocalDate"(optional) The UNTIL part from iCalendar. This MUST NOT be included if a *count* property is specified. Note, as in iCalendar, this date is presumed to be in the time-zone specified in *timeZone*. It is not a UTC time.

4.3.2. recurrenceOverrides

Type: "LocalDate[PatchObject|null]|null"

A map of the recurrence-ids (the date-time of the start of the occurrence) to either "null", to indicate the occurrence should be deleted, or an object of patches to apply to the generated occurrence object.

If the recurrence-id does not match an expanded start date from a recurrence rule, it is to be treated as an additional occurrence (like an RDATE from iCalendar). The patch object may often be empty in this case.

By default, an occurrence inherits all properties from the main object except the start (or due) date-time, which is shifted to the new start time. However, individual properties of the occurrence can be modified by a patch, or multiple patches.

A pointer in the PatchObject MUST NOT start with one of the following prefixes; any patch with such a key MUST be ignored:

- @type

- uid

- relatedTo
- prodId
- method
- isAllDay
- recurrenceRule
- recurrenceOverrides
- replyTo

4.4. Sharing and scheduling properties

4.4.1. priority

Type: "Number"(defaults to "0" if omitted)

Specifies a priority for the calendar object. This may be used as part of scheduling systems to help resolve conflicts for a time period.

The priority is specified as an integer in the range 0 to 9. A value of 0 specifies an undefined priority. A value of 1 is the highest priority. A value of 2 is the second highest priority. Subsequent numbers specify a decreasing ordinal priority. A value of 9 is the lowest priority. Other integer values are reserved for future use.

4.4.2. freeBusyStatus

Type: "String"(defaults to "busy" if omitted)

Specifies how this property should be treated when calculating free-busy state. The value MUST be one of:

- ""free"": The object should be ignored when calculating whether the user is busy.
- ""busy"": The object should be included when calculating whether the user is busy.

4.4.3. privacy

Type: "String"(defaults to "public" if omitted)

Calendar objects are normally collected together and may be shared with other users. The privacy property allows the object owner to indicate that it should not be shared, or should only have the time information shared but the details withheld. Enforcement of the restrictions indicated by this property are up to the implementations.

This property MUST NOT affect the information sent to scheduled participants; it is only interpreted when the object is shared as part of a shared calendar.

The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value. Vendor specific values MUST be prefixed with a domain name controlled by the vendor, e.g. "example.com/topsecret". Any value the client or server doesn't understand should be preserved but treated as equivalent to "private".

- "public": The full details of the object are visible to those whom the object's calendar is shared with.

- "private": The details of the object are hidden; only the basic time and metadata is shared. Implementations MUST ensure the following properties are stripped when the object is accessed by a sharee:

  * title

  * description

  * htmlDescription

  * locations

  * links

  * locale

  * localizations

  * participants

  * replyTo

  In addition, any patches in "recurrenceOverrides" whose key is prefixed with one of the above properties MUST be stripped.

- "secret": The object is hidden completely (as though it did not exist) when the calendar is shared.

4.4.4. replyTo

Type: "String[String]|null"

Represents methods by which a participant may RSVP to the organizer of the calendar object. The keys in the property value are the available methods. The value is a URI to use that method. Future methods may be defined in future specifications; a calendar client MUST ignore any method it does not understand.

The following methods are defined:

- "imip": The organizer accepts an iMIP [RFC6047] response. The value MUST be a "mailto:" URI.

- "web": There is a web page where the user may submit an RSVP using their browser. The value MUST be an "http:" or "https:" URI Template ([RFC6570]) in level 1 format. The template MAY contain variables that MUST be expanded from the JSCalendar object as defined in table Table 1. Calendar clients SHOULD be prepared to handle authentication requests from the respective web page and for the participant email, but this specification does not mandate any specific mechanism.

Variable	Expand to
email	The *email* property value of the replying *Participant* object.
uid	The *uid* property value of the JSCalendar object.
sequence	The *sequence* property value of the JSCalendar object.
recurrenceId	The recurrence-id when replying for a single occurrence of a recurring JSCalendar object. The value is the date-time of the non-overridden start as determined by expanding the *recurrenceRule* of the JSCalendar object.

Table 1: replyTo URI Template variables

4.4.5. participants

Type: "String[Participant]|null"

A map of participant ids to participants, describing their participation in the calendar object. A participant id may be any string and need only be unique to this calendar object; the email address of the participant is a good choice.

A *Participant* object has the following properties. Properties are mandatory unless marked otherwise:

- *name*: "String" The display name of the participant (e.g. "Joe Bloggs").
- *email*: "String" The email address for the participant.
- *kind*: "String"(optional, defaults to "unknown") What kind of entity this participant is.

  This MUST be either one of the following values, registered in a future RFC, or a vendor-specific value. Any value the client or server doesn't understand should be treated the same as "unknown".

  * "individual": a single person
  * "group": a collection of people invited as a whole
  * "resource": a non-human resource, e.g. a projector
  * "location": a physical location involved in the calendar object that needs to be scheduled, e.g. a conference room.
  * "unknown": no information is available about the kind of this participant.

- *roles*: "String[]" A list of roles that this participant fulfils.

  At least one value MUST be specified for the participant. This MUST be either one of the following values, registered in a future RFC, or a vendor-specific value. Any value the client or server doesn't understand should be preserved but ignored.

  * "owner": The participant is an owner of the object, and allowed to make alterations to any part of it.

  * "attendee": The participant is an attendee of the calendar object. Attendees are only allowed to alter their own participation.

  * "chair": The participant is in charge of the calendar object when it occurs.

- *locationId*: "String|null"(optional, defaults to "null") The location at which this participant is expected to be attending.

  If the value does not correspond to any location id in the locations property of the instance, this MUST be treated the same as if the participant's locationId were specified as null.

- *rsvpResponse*: "String"(optional, defaults to "needs-action") The RSVP response, if any, of this participant.

  The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:

  * "needs-action": No status yet set by the participant.

  * "accepted": The invited participant will participate.

  * "declined": The invited participant will not participate.

  * "tentative": The invited participant may participate.

- *participation*: "String"(optional, defaults to "required") The required participation of this participant.

  The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value. Any value the client or server doesn't understand should be treated the same as "required".

  * "non-participant": Indicates a participant who is copied for information purposes only.

  * "optional": Indicates a participant whose participation is optional.

  * "required": Indicates a participant whose participation is required.

- *rsvpWanted*: "Boolean"(optional, defaults to "false") If true, the organizer is expecting the participant to notify them of their status.

- *scheduleSequence*: "Number"(optional, defaults to "0") The sequence number of the last response from the participant. If defined, this MUST be a non-negative integer.

  This can be used to determine whether the partcipant has sent a new RSVP following significant changes to the calendar object, and to determine if future responses are responding to a current or older view of the data.

- *scheduleUpdated*: "UTCDate|null"(optional, defaults to "null") The *updated* property of the last iMIP response from the participant.

  This can be compared to the *updated* timestamp in future iMIP responses to determine if the response is older or newer than the current data.

- *invitedBy*: "String|null"(optional, defaults to "null") The participant id of the participant who invited this one, if known.

- *delegatedTo*: "String[]|null"(optional, defaults to "null") A list of participant ids of participants that this participant has delegated their participation to. This MUST be omitted if none (rather than an empty array).

- *delegatedFrom*: "String[]|null"(optional, defaults to "null") A list of participant ids that this participant is acting as a delegate for. This MUST be omitted if none (rather than an empty array).

- *memberOf*: "String[]|null"(optional, defaults to "null") A list of group participants that were invited to this calendar object, which caused this participant to be invited due to their membership of the group(s). This MUST be omitted if none (rather than an empty array).

- *linkIds*: "String[]|null"(optional, defaults to "null") Links to more information about this participant, for example in vCard format. If a given value does not correspond to any link id in the links property of the instance, this id MUST be ignored. This MUST be omitted if none (rather than an empty array).

4.5. Alerts properties

4.5.1. useDefaultAlerts

Type: "Boolean" (defaults to "false" if omitted)

If "true", use the user's default alerts and ignore the value of the *alerts* property. Fetching user defaults is dependent on the API from which this JSCalendar object is being fetched, and is not defined in this specification. If an implementation cannot determine the user's default alerts, or none are set, it MUST process the alerts property as if useDefaultAlerts is set to "false".

4.5.2. alerts

Type: "String[Alert]|null"

A map of alert ids to Alert objects, representing alerts/reminders to display or send the user for this calendar object. An alert id may be any string and need only be unique to this calendar object, although a globally unique id is a practical choice (also see Section 4.1.2)).

An *Alert* Object has the following properties:

- *relativeTo*: "String" (optional, defaults to "before-start") Specifies where the offset is relative to for the alarm to trigger. The value MUST be one of:
  - "before-start"
  - "after-start"
  - "before-end"
  - "after-end"
- *offset*: "Duration" The offset from the start and end/due of the calendar object to fire the alert. If the calendar object does not define a time-zone, the user's default time-zone SHOULD be used when determining the offset, if known. Otherwise, the time-zone to use is implementation specific.
- *action*: "DisplayAction|EmailAction|UnknownAction"

  Describes how to alert the user.

  A *DisplayAction* means a message (which is service dependent, but SHOULD include the title and start or due time of the calendar object) SHOULD be shown to the user on any client connected to

this account at the specified time. How this message is formatted (and any sound or other method of drawing the user's attention) is client specific. It has the following properties:

* *type*: "String" The value MUST be "display".

* *acknowledged*: "UTCDate|null " (optional)

  When the user has permanently dismissed the alert the client MUST set this to the current time in UTC. Other clients which sync this property can then automatically dismiss or suppress duplicate alerts (alerts with the same alert id that triggered on or before this date-time).

  For a recurring calendar object, the *acknowledged* property of the parent object MUST be updated, unless the alert is already overridden in *recurrenceOverrides*.

* *snoozed*: "UTCDate|null" (optional)

  If the user temporarily dismisses the alert, this is the UTC date-time after which it should be reshown. Clients displaying this alert SHOULD hide it if the snoozed property is updated to a time in the future. When that time is reached, the alert SHOULD be reshown unless acknowledged is now after the original trigger time.

  Setting this property on an instance of a recurring calendar object MUST update the alarm on the master object, unless the respective instance already is defined in "recurrenceOverrides". It MUST NOT generate an override for the sole use of snoozing an alarm.

* *mediaLinkIds*: "String[]|null " (optional)

  A list of link identifiers in the JSCalendar object *links* property. Clients SHOULD play one or more of the link contents that are supported by the client implementation and are appropriate for the given device and user context.

An *EmailAction* means an email SHOULD be sent as specified in the object at the specified time. It has the following properties:

* *type*: "String" The value MUST be "email".

* *to*: "Emailer[]" An array of name/email objects to send the alert to.

An *Emailer* object has the following properties:

+ name: String The name of the recipient. If not known, clients SHOULD use the empty string.

+ email: String The email address of the recipient.

* *subject*: "String" (optional) The subject to use for the email. If omitted, this is implementation specific, but the server SHOULD try to choose an appropriate subject, e.g. by including the summary.

* *textBody*: "String" (optional) The plain-text body to use for the email. If omitted, the body of the email is implementation specific, but the server SHOULD include all pertinent details about the calendar object, such as summary, location and start time.

* *htmlBody*: "String" (optional) The HTML body to use for the email, with rich-media content processed as for the *htmlDescription* property of the JSCalendar object (see Section 4.2.3), e.g. all CID URLs MUST be embedded in the generated alert email HTML body, or the *htmlBody* property ignored completely. If the textBody property of this alert action is not set, the server SHOULD generate a plain-text version from the HTML body and include it in a "multipart/alternative" MIME message.

* *attachments*: "String[]|null" (optional) A list of link identifiers in the JSCalendar object links property. Included attachments SHOULD be embedded in the MIME message with the "Content-Disposition" header value set to "attachment" (see [RFC2183]). Implementations MAY refuse to include one or more attachments when building an alert email, in which case they MUST ignore the contents of the *attachments* property (e.g. they MUST NOT include a subset of attachments).

An *UnknownAction* object is an object that contains a *type * property whose value is not "email" or "string", plus zero or more other properties. This is for compatibility with client extensions and future RFCs. The client or server SHOULD NOT trigger any type of alert for action types they do not understand, but MUST preserve them.

4.6. Multilingual properties

4.6.1. localizations

Type: "String[PatchObject]|null"

A map of [RFC5646] language tags to patch objects, which localise the calendar object into the locale of the respective language tag.

See the description of PatchObject (Section 3.2.4) for the structure of the PatchObject. The patches are applied to the top-level object and MUST only patch the following properties:

- title
- description
- htmlDescription
- keywords
- Location name
- Location description
- Link title
- EmailAction alert subject
- EmailAction alert textBody
- EmailAction alert htmlBody
- any Link properties entry except the "display" field
- any UnknownAction alert property except the "type" field
- any unknown or vendor-specific property (if not defined otherwise in a future RFC or vendor-specific extension).

5. Type-specific JSCalendar properties

5.1. JSEvent properties

In addition to the common JSCalendar object properties (Section 4) a JSEvent has the following properties:

5.1.1. start

Type: "LocalDate" e.g. "2015-09-02T00:00:00"

The date/time the event would start in the event’s time-zone.

A valid JSEvent MUST include this property.

5.1.2. timeZone

Type: "String|null"

The IANA Time Zone Database [4] name for the time-zone the event is scheduled in, or "null" for floating time. If omitted, this MUST be presumed to be "null" (i.e. floating time).

5.1.3. duration

Type: "Duration", e.g. "P2DT3H" (Defaults to "P0D" if omitted)

The zero or positive duration of the event in absolute time (i.e. in UTC time; ignoring DST shifts). To get the end date in the event time-zone, convert start into UTC, then add the duration, then convert the result into the appropriate time-zone.

A JSEvent MAY be end in a different time-zone (e.g. a plane flight crossing time-zones). In this case, the JSEvent MUST specify the end time-zone in a *location* property value that defines its *rel* to be "end" and the end time-zone in its *timeZone* property.

5.1.4. isAllDay

Type: "Boolean" (optional, defaults to "false")

Specifies if the event an all day event, such as a birthday or public holiday.

If *isAllDay* is true, then the following restrictions apply:

- the *start* property MUST have a time component of "T00:00:00".
- the *duration* property MUST only include a day component.

Note that all-day events MAY be bound to a specific time-zone, as defined by the *timeZone* property.

5.1.5. status

Type: "String"

The scheduling status (Section 4.4) of a JSEvent defaults to "confirmed" if omitted.

If set, it MUST be one of:

- "confirmed": Indicates the event is definite.
- "cancelled": Indicates the event is cancelled.
- "tentative": Indicates the event is tentative.

5.2. JSTask properties

In addition to the common JSCalendar object properties (Section 4) a JSTask has the following properties:

5.2.1. due

Type: "LocalDate|null" e.g. "2015-09-02T00:00:00"

The date/time the task is due in the task’s time-zone.

5.2.2. start

Type: "LocalDate|null" e.g. "2015-09-02T00:00:00"

The date/time the task should start in the task’s time-zone.

5.2.3. timeZone

Type: "String|null"

The IANA Time Zone Database name for the time-zone the task is scheduled in, or "null" for floating time. If omitted, this MUST be presumed to be "null" (i.e. floating time).

5.2.4. estimatedDuration

Type: "Duration|null", e.g. "P2DT3H"

Specifies the estimated positive duration of time the task takes to complete.

If the *start* and *due* properties are set, the estimated duration SHOULD be less than or equal to the time interval between these properties.

5.2.5. completed

Type: "UTCDate|null", e.g. "2016-06-13T12:00:00Z"

Specifies the date/time the task was completed.

If the task is recurring and has future instances, a client may want to denote a specific task recurrence as completed but leave other instances as uncompleted. One way to achieve this is by overriding the completed property in the task *recurrenceOverrides*. However, this could produce a long list of completion times for regularly recurring tasks. An alternative approach is to split the JSTask into a current, single instance of JSTask with this instance completion time and a future recurring instance. Also see the definition of the *relatedTo* on splitting.

5.2.6. isAllDay

Type: "Boolean" (optional, defaults to "false")

Specifies if the task is an all day task.

If *isAllDay* is true, then the *start* and *due* properties MUST have a time component of "T00:00:00". Note that the *estimatedDuration* property MAY contain a non-zero time duration. All-day tasks MAY be bound to a specific time-zone, as defined by the *timeZone* property.

5.2.7. progress

In addition to the common properties of a *Participant* object (Section 4.4.5), a Participant within a JSTask supports the following property:

- *progress*: "ParticipantProgress|null" The progress of the participant for this task, if known.

A *ParticipantProgress* object has the following properties:

- *status*: "String" Describes the completion status of the participant’s progress.

The value MUST be at most one of the following values, registered in a future RFC, or a vendor-specific value:

* "completed": The progress of this participant is complete.

* "in-process": The progress of this participant is in process.

- *timestamp*: "UTCDate" Describes the latest time when the participant progress got updated.

5.2.8. status

Type: "String"

The scheduling status (Section 4.4) of a JSTask defaults to "needs-action" if omitted.

If set, it MUST be one of:

- "needs-action": Indicates the task needs action.
- "completed": Indicates the task is completed. If this value is set, then the timestamp in the *completed* property MUST NOT be null.
- "in-process": Indicates the task is in process.
- "cancelled": Indicates the task is cancelled.

5.3. JSGroup properties

JSGroup supports the following JSCalendar properties (Section 4):

- @type
- uid
- created
- updated
- categories
- keywords
- name
- description

- o htmlDescription
- o color
- o links

as well as the following JSGroup-specific properties:

5.3.1. entries

Type: "(JSTask|JSEvent)[]|null"

A list of group members. The list MAY contain multiple object types and implementations MUST ignore entries of unknown type. The property value MUST either be "null" or the list MUST NOT be empty.

5.3.2. source

Type: "String|null" (optional, default is "null")

The source from which updated versions of this group may be retrieved from. If the value is not "null", it MUST be a URI.

6. Conversion from and to iCalendar

This section specifies which JSCalendar properties can be mapped from and to iCalendar format. Implementations SHOULD follow these conversion guidelines. Still, JSCalendar does not restrict itself to iCalendar and conversion between these two formats MAY be lossy.

6.1. JSEvent

The iCalendar counterpart to *JSEvent* is the VEVENT component type [RFC5545]. A VEVENT component that is a direct child of a VCALENDAR component is equivalent to a standalone JSEvent. A VEVENT component *within* a VEVENT maps to the entries of the JSEvent *recurrenceOverrides* property.

Property	iCalendar counterpart
isAllDay	True, if the type of the DTSTART property in iCalendar is DATE. When translating from JSCalendar the iCalendar DTSTART property is of DATE value type, if the *isAllDay* property is set to true and the *timeZone* property is null.
start	Corresponds to the DTSTART property in iCalendar. Note that time-zone information is stored separately in JSEvent.
timeZone	Corresponds to the TZID part of the DTSTART property in iCalendar. If the event has a different end time-zone to start time-zone, this should be added as a JSCalendar *location* with just a *timeZone* property and "rel="end"".
duration	Corresponds to the DURATION or DSTART+DTEND properties in iCalendar.

Table 2: Translation between JSEvent and iCalendar

6.2. JSTask

The iCalendar counterpart to *JSTask* is the VTODO component type [RFC5545]. A VTODO component that is a direct child of a VCALENDAR component is equivalent to a standalone JSTask. A VTODO component *within* a master VTODO maps to the entries of the JSTask *recurrenceOverrides* property.

Property	iCalendar counterpart
isAllDay	True, if the type of the DTSTART property in iCalendar is DATE. When translating from JSCalendar the iCalendar DTSTART property is of DATE value type, if the *isAllDay* property is set to true and the *timeZone* property is null.
due	Corresponds to the DUE and DTSTART+DURATION properties in iCalendar. When mapping iCalendar VTODOs with DTSTART+DURATION, the due date is the result of adding DURATION to DTSTART in the DTSTART time-zone.
start	Corresponds to the DTSTART property in iCalendar.
timeZone	Corresponds to the TZID part of the DTSTART/DUE properties in iCalendar. If the task has a different end time-zone to start or due time-zone, this should be added as a JSCalendar *location* with just a *timeZone* property and "rel="end"".
estimatedDuration	Corresponds to the ESTIMATED-DURATION iCalendar property. *NON-STANDARD*: this property is currently non-standard, see [draft-apthorp-ical-tasks].
completed	Maps to the COMPLETED iCalendar property.
progress	Corresponds to the PARTSTAT and COMPLETED properties in iCalendar.

Table 3: Translation between JSTask and iCalendar

6.3. JSGroup

A JSGroup converts to a iCalendar VCALENDAR containing VEVENT or VTODO components.

Property	iCalendar counterpart
entries	The VEVENT and VTODO components within a top-level VCALENDAR component.
source	Corresponds to the SOURCE property in iCalendar.

Table 4: Translation between JSGroup and iCalendar

6.4. Common properties

Property	iCalendar counterpart
alerts	An *Alert* corresponds to the VALARM component in iCalendar, where the *action* is determined by the iCalendar ACTION property value (e.g., a "DISPLAY" property indicates that the JSCalendar Alert action is a *DisplayAction* and similarly an iCalendar "EMAIL" value for *EmailAction* action). The *relativeTo* and *offset* properties corresponds to the iCalendar TRIGGER property. *NON-STANDARD*: The iCalendar properties for JSCalendar Alert actions are non-standard, see [draft-daboo-valarm-extensions].
categories	Corresponds to the STRUCTURED-CATEGORY property in iCalendar, see. *NON-STANDARD*: this property is currently non-standard, see [draft-ietf-calext-ical-relations].
color	Corresponds to the COLOR property in iCalendar, as specified in [RFC7986].
created	Corresponds to the CREATED property in iCalendar.
description	Corresponds to the DESCRIPTION property in iCalendar.
htmlDescription	There is no direct equivalent in iCalendar. If the *description* is empty, implementations SHOUD store a plain text

	version of *htmlDescription* in iCalendar DESCRIPTION.
freeBusyStatus	Corresponds to the TRANSP property in iCalendar.
keywords	Corresponds to the CATEGORIES property in iCalendar, as specified in [RFC7986].
links	Corresponds to the ATTACH ([RFC5545]) and IMAGE iCalendar properties ([RFC7986]).
locale	Corresponds to the LANGUAGE parameter in iCalendar, which is added to individual properties. When converting from iCalendar, one language must be picked as the main locale for the object, and all properties in other languages moved to the localizations JSEvent property.
localizations	Corresponds to the LANGUAGE parameter in iCalendar, which is added to individual properties. When converting from iCalendar, one language must be picked as the main locale for the object, and all properties in other languages moved to the localizations JSEvent property.
locations	See Section 6.5.
method	Corresponds to the METHOD property in iCalendar.
participants	See Section 6.5.
priority	Corresponds to the PRIORITY property in iCalendar.
privacy	Corresponds to the CLASS property in iCalendar.
prodId	Corresponds to the PRODID property in iCalendar.
recurrenceOverrides	Corresponds to the RDATE and EXDATE properties in iCalendar, plus VEVENT (for JSEvent) or VTODO (for JSTask) instances with a recurrence-id.

recurrenceRule	Corresponds to the RRULE property in iCalendar. See the property definition at section Section 4.3.1 how to map a RRULE value.
relatedTo	Corresponds to the RELATED-TO property in iCalendar.
replyTo	A *replyTo* property of type "imip" corresponds to the email address of the ORGANIZER property in iCalendar. There is no iCalendar representation for the "web" type.
sequence	Corresponds to the SEQUENCE property in iCalendar.
status	Corresponds to the STATUS property in iCalendar (converted to lower-case).
title	Corresponds to the SUMMARY property in iCalendar.
uid	Corresponds to the UID property in iCalendar.
updated	Corresponds to the DTSTAMP and LAST-MODIFIED properties in iCalendar. (These are only different in the iTIP case, and the difference is not actually useful.)

Table 5: Translation between JSCalendar and iCalendar

6.5. Locations and participants

Both JSCalendar participants and locations have counterparts in iCalendar but provide richer representation.

The following table outlines translation of JSCalendar participants. Where iCalendar has distinct properties for ORGANIZER and ATTENDEE, these are merged in JSCalendar into the Participant object type.

Property	iCalendar counterpart
name	the CN parameter
kind	the CUTYPE parameter
rsvpResponse	the PARTSTAT parameter
role	the ORGANIZER and ATTENDEE properties. Owners map to the iCalendar ORGANIZER property, where mapping multiple owners to iCalendar is implementation-specific but MUST be consistent across mappings of the same object.
participation	the ROLE parameter
locationId	the JSCalendar identifier of a mapped CONFERENCE property that has the MODERATOR feature defined in its FEATURE parameter values. If multiple such CONFERENCE properties are defined in iCalendar, then the one with the most interactive features is chosen (VIDEO over AUDIO over CHAT over anything else).
rsvpWanted	the RSVP parameter
delegatedTo	the DELEGATED-TO parameter
delegatedFrom	the DELEGATED-FROM parameter
memberOf	the MEMBER parameter
scheduleSequence	the SEQUENCE property of the participant's latest iMIP message
scheduleUpdated	the DTSTAMP property of the participant's latest iMIP message

Table 6: Translation of Participant between JSCalendar and iCalendar

For JSCalendar locations, the iCalendar counterparts are the [RFC5545] LOCATION and the extended iCalendar [RFC7986] CONFERENCE properties.

An iCalendar LOCATION property becomes a JSCalendar Location with just a description property. CONFERENCE property values in iCalendar

map to locations with *rel* type "virtual". The location *feature* property value corresponds to the extended iCalendar FEATURE property parameter values defined in [RFC7986]. Both iCalendar PHONE and AUDIO features map to the "audio" feature and the FEED parameter value is omitted. See the mapping for *locationId* in Table 6 on how to map CONFERENCE properties that contain the MODERATOR feature.

6.6. Unknown properties

Both JSCalendar and iCalendar calendar objects may contain properties that are not expressible in the other format. This specification does not mandate how to preserve these properties. Instead, it leaves negotiation on how to treat unknown properties to client and server implementations and their protocol used to exchange calendar objects.

Two notable options to represent and preserve arbitrary iCalendar object properties in JSCalendar are:

- *JCal*: Define iCalendar properties in JCal format ([RFC7265]) in a vendor-specific property of the JCalendar object. The JCal-formatted value may either only contain iCalendar properties that were not mapped to JSCalendar properties, or contain the complete iCalendar object representation.

- *Alternate link*: Define an alternate link (Section 4.2.5) value pointing to the iCalendar representation of the JSCalendar object. E.g. the alternative representation of a VEVENT would be represented as a link with rel "alternate" and type "text/calendar;component=VEVENT".

7. JSCalendar object examples

The following examples illustrate several aspects of the JSCalendar data model and format. The examples may omit mandatory or additional properties, which is indicated by a placeholder property with key "...". While most of the examples use calendar event objects, they are also illustrative for tasks.

7.1. Simple event

This example illustrates a simple one-time event. It specifies a one-time event that begins on January 15, 2018 at 1pm New York local time and ends after 1 hour.

```
{
  "@type": "jsevent",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f1",
  "updated": "2018-01-15T18:00:00Z",
  "title": "Some event",
  "start": "2018-01-15T13:00:00",
  "timeZone": "America/New_York",
  "duration": "PT1H"
}
```

7.2. Simple task

This example illustrates a simple task for a plain to-do item.

```
{
  "@type": "jstask",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
  "updated": "2018-01-15T18:00:00Z",
  "title": "Do something"
}
```

7.3. Simple group

This example illustrates a simple calendar object group that contains an event and a task.

```
{
  "@type": "jsgroup",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc343",
  "updated": "2018-01-15T18:00:00Z",
  "name": "A simple group",
  "entries": [
    {
      "@type": "jsevent",
      "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f1",
      "updated": "2018-01-15T18:00:00Z",
      "title": "Some event",
      "start": "2018-01-15T13:00:00",
      "timeZone": "America/New_York",
      "duration": "PT1H"
    },
    {
      "@type": "jstask",
      "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
      "updated": "2018-01-15T18:00:00Z",
      "title": "Do something"
    }
  ]
}
```

7.4. All-day event

This example illustrates an event for an international holiday. It specifies an all-day event on April 1 that occurs every year since the year 1900.

```
{
  "...": "",
  "title": "April Fool’s Day",
  "isAllDay": true,
  "start": "1900-04-01T00:00:00",
  "duration": "P1D",
  "recurrenceRule": {
    "frequency": "yearly"
  }
}
```

7.5. Task with a due date

This example illustrates a task with a due date. It is a reminder to buy groceries before 6pm Vienna local time on January 19, 2018. The calendar user expects to need 1 hour for shopping.

```
{
  "...": "",
  "title": "Buy groceries",
  "due": "2018-01-19T18:00:00",
  "timeZone": "Europe/Vienna",
  "estimatedDuration": "PT1H"
}
```

7.6. Event with end time-zone

This example illustrates the use of end time-zones by use of an international flight. The flight starts on April 1, 2018 at 9am in Berlin local time. The duration of the flight is scheduled at 10 hours 30 minutes. The time at the flights destination is in the same time-zone as Tokyo. Calendar clients could use the end time-zone to display the arrival time in Tokyo local time and highlight the time-zone difference of the flight.

```
{
  "...": "",
  "title": "Flight XY51 from FRA to NRT",
  "start": "2018-04-01T09:00:00",
  "timeZone": "Europe/Berlin",
  "duration": "PT10H30M",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "rel": "end",
      "timeZone": "Asia/Tokyo"
    }
  }
}
```

7.7. Floating-time event (with recurrence)

This example illustrates the use of floating-time. Since January 1, 2018, a calendar user blocks 30 minutes every day to practice Yoga at 7am local time, in whatever time-zone the user is located on that date.

```
{
  "...": "",
  "title": "Yoga",
  "start": "2018-01-01T07:00:00",
  "duration": "PT30M",
  "recurrenceRule": {
    "frequency": "daily"
  }
}
```

7.8. Event with multiple locations and localization

This example illustrates an event that happens at both a physical and a virtual location. Fans can see a live convert on premises or online. The event title and descriptions are localized. (Note: the localization of the event description contains an UTF-8 encoded German Umlaut. This character may have been replaced with ASCII characters in the plain-text rendering of this RFC document)

```
{
  "...": "",
  "title": "Live from Music Bowl: The Band",
  "description": "Go see the biggest music event ever!",
  "locale": "en",
  "start": "2018-07-04T17:00:00",
  "timeZone": "America/New_York",
  "duration": "PT3H",
  "locations": {
    "9366e041-bb4c-4aa4-b249-b4657cab925c": {
      "name": "The Music Bowl",
      "description": "Music Bowl, Central Park, New York",
      "coordinates": "geo:40.7829,73.9654"
    },
    "6f3696c6-1e07-47d0-9ce1-f50014b0041a": {
      "name": "Free live Stream from Music Bowl",
      "rel": "virtual",
      "features": [
        "video",
        "audio",
        "chat"
      ],
      "uri": "https://stream.example.com/the_band_2018"
    }
  },
  "localizations": {
    "de": {
      "title": "Live von der Music Bowl: The Band!",
      "description": "Schau dir das groesste Musikereignis an!",
      "locations/6f3696c6-1e07-47d0-9ce1-f50014b0041a/name":
                            "Gratis Live-Stream aus der Music Bowl"
    }
  }
}
```

7.9. Recurring event with overrides

This example illustrates the use of recurrence overrides. A math course at a University is held for the first time on January 8, 2018 at 9am London time and occurs every week until June 25, 2018. Each lecture lasts for one hour and 30 minutes and is located at the Mathematics department. This event has exceptional occurrences: at the last occurrence of the course is an exam, which lasts for 2 hours and starts at 10am. Also, the location of the exam differs from the usual location. On April 2, May 7 and May 28 no course is held.

```
{
  "...": "",
  "title": "Calculus I",
  "start": "2018-01-08T09:00:00",
  "timeZone": "Europe/London",
  "duration": "PT1H30M",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "title": "Math lab room 1",
      "description": "Math Lab I, Department of Mathematics"
    }
  },
  "recurrenceRule": {
    "frequency": "weekly",
    "until": "2018-06-25T09:00:00"
  },
  "recurrenceOverrides": {
    "2018-04-02T09:00:00": null,
    "2018-05-07T09:00:00": null,
    "2018-05-28T09:00:00": null,
    "2018-06-25T09:00:00": {
      "title": "Calculus I Exam",
      "start": "2018-06-25T10:00:00",
      "duration": "PT2H",
      "locations": {
        "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
          "title": "Big Auditorium",
          "description": "Big Auditorium, Other Road"
        }
      }
    }
  }
}
```

7.10. Recurring event with participants

This example illustrates scheduled events. A team meeting occurs every week since January 8, 2018 at 9am Johannesburg time. The event owner also chairs the event. Participants meet in a virtual meeting room. An attendee has accepted the invitation, but on March 8, 2018 he is unavailable and declined participation for this occurrence.

```
{
  "...": "",
  "title": "FooBar team meeting",
  "start": "2018-01-08T09:00:00",
  "timeZone": "Africa/Johannesburg",
  "duration": "PT1H",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "title": "ChatMe meeting room",
      "rel": "virtual",
      "features": [
        "audio",
        "chat",
        "video"
      ],
      "uri": "https://chatme.example.com?id=1234567"
    }
  },
  "recurrenceRule": {
    "frequency": "weekly"
  },
  "replyTo": {
    "imip": "zoe@foobar.example.com"
  },
  "participants": {
    "tom@foobar.example.com": {
      "name": "Tom Tool",
      "email": "tom@foobar.example.com",
      "rsvpResponse": "accepted",
      "roles": [
        "attendee"
      ]
    },
    "zoe@foobar.example.com": {
      "name": "Zoe Zelda",
      "email": "zoe@foobar.example.com",
      "rsvpResponse": "accepted",
      "roles": [
        "owner",
        "chair"
```

```
        ]
      },
      "...": ""
    },
    "recurrenceOverrides": {
      "2018-03-08T09:00:00": {
        "participants/tom@foobar.example.com/rsvpResponse": "declined"
      }
    }
  }
```

8. Security Considerations

The use of JSON as a format does have its own inherent security risks as discussed in Section 12 of [RFC8259]. Even though JSON is considered a safe subset of JavaScript, it should be kept in mind that a flaw in the parser processing JSON could still impose a threat, which doesn’t arise with conventional iCalendar data.

With this in mind, a parser for JSON data aware of the security implications should be used for the format described in this document. For example, the use of JavaScript’s "eval()" function is considered an unacceptable security risk, as described in Section 12 of[RFC8259]. A native parser with full awareness of the JSON format should be preferred.

9. IANA Considerations

This document amends the "application/calendar" MIME media type defined in [RFC7265].

New optional parameter: "type" with value being one of "jsevent", "jstask", "jsgroup". The parameter MUST NOT occur more than once.

10. Acknowledgments

The authors would like to thank the members of CalConnect for their valuable contributions. This specification originated from the work of the API technical committee of CalConnect, the Calendaring and Scheduling Consortium.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2183] Troost, R., Dorner, S., and K. Moore, Ed., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, DOI 10.17487/RFC2183, August 1997, <https://www.rfc-editor.org/info/rfc2183>.

[RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, DOI 10.17487/RFC2392, August 1998, <https://www.rfc-editor.org/info/rfc2392>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <https://www.rfc-editor.org/info/rfc4791>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/info/rfc5545>.

[RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <https://www.rfc-editor.org/info/rfc5546>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations (’geo’ URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <https://www.rfc-editor.org/info/rfc5870>.

[RFC6047] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 6047, DOI 10.17487/RFC6047, December 2010, <https://www.rfc-editor.org/info/rfc6047>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

[RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", RFC 7265, DOI 10.17487/RFC7265, May 2014, <https://www.rfc-editor.org/info/rfc7265>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[RFC7529] Daboo, C. and G. Yakushev, "Non-Gregorian Recurrence Rules in the Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 7529, DOI 10.17487/RFC7529, May 2015, <https://www.rfc-editor.org/info/rfc7529>.

[RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <https://www.rfc-editor.org/info/rfc7986>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288,
DOI 10.17487/RFC8288, October 2017,
<https://www.rfc-editor.org/info/rfc8288>.

11.2. Informative References

[draft-apthorp-ical-tasks]
"Task Extensions to iCalendar",
<https://tools.ietf.org/html/draft-apthorp-ical-tasks>.

[draft-daboo-valarm-extensions]
"VALARM Extensions for iCalendar",
<https://tools.ietf.org/html/
draft-daboo-valarm-extensions>.

[draft-ietf-calext-ical-relations]
"Support for iCalendar Relationships",
<https://tools.ietf.org/html/
draft-ietf-calext-ical-relations>.

11.3. URIs

[1] https://www.w3.org/TR/html/

[2] https://www.iana.org/assignments/link-relations/link-relations.xhtml

[3] https://www.w3.org/TR/2011/REC-css3-color-20110607/#svg-color

[4] http://www.iana.org/time-zones

Authors' Addresses

Neil Jenkins
FastMail
PO Box 234
Collins St West
Melbourne VIC 8007
Australia

Email: neilj@fastmailteam.com
URI: https://www.fastmail.com

Robert Stepanek
FastMail
PO Box 234
Collins St West
Melbourne VIC 8007
Australia

Email: rsto@fastmailteam.com
URI: https://www.fastmail.com