Abstract

This document defines a set of extensions to the iCalendar VALARM component to enhance use of alarms and improve interoperability between clients and servers.

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 September 6, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

The iCalendar [RFC5545] specification defines a set of components used to describe calendar data. One of those is the "VALARM" component which appears as a sub-component of "VEVENT" and "VTODO" components. The "VALARM" component is used to specify a reminder for an event or task. Different alarm actions are possible, as are different ways to specify how the alarm is triggered.

As iCalendar has become more widely used and as client-server protocols such as CalDAV [RFC4791] have become more popular, several issues with "VALARM" components have arisen. Most of these relate to the need to extend the existing "VALARM" component with new properties and behaviors to allow clients and servers to accomplish specific tasks in an interoperable manner. For example, clients typically need a way to specify that an alarm has been dismissed by a calendar user, or has been "snoozed" by a set amount of time. To date, this has been done through the use of custom "X-" properties specific to each client implementation, leading to poor interoperability.

This specification defines a set of extensions to "VALARM" components to cover common requirements for alarms not currently addressed in
iCalendar. Each extension is defined in a separate section below. For the most part, each extension can be supported independently of the others, though in some cases one extension will require another. In addition, this specification describes mechanisms by which clients can interoperably implement common features such as "snoozing".

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 BCP 14 [1] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

When XML element types in the namespaces "DAV:" and "urn:ietf:params:xml:ns:caldav" are referenced in this document outside of the context of an XML fragment, the string "DAV:" and "CALDAV:" will be prefixed to the element type names respectively.

3. Extensible syntax for VALARM

Section 3.6.6 of [RFC5545] defines the syntax for "VALARM" components and properties within them. However, as written, it is hard to extend this by adding, e.g., a new property common to all types of alarm. Since many of the extensions defined in this document need to extend the base syntax, an alternative form for the base syntax is defined here, with the goal of simplifying specification of the extensions.

A "VALARM" calendar component is re-defined by the following notation:

alarmcext  = "BEGIN" ":" "VALARM" CRLF
    alarmprop
        "END" ":" "VALARM" CRLF

alarmprop  = *(        ; the following are REQUIRED,
    action / trigger /          ; but MUST NOT occur more than once
    ; one set of action properties MUST be
    ; present and MUST match the action specified
    ; in the ACTION property
    actionprops / )
x-prop / iana-prop
)

actionprops = audiopropext / disppropext / emailpropext

audiopropext = *

; 'duration' and 'repeat' are both OPTIONAL,
; and MUST NOT occur more than once each,
; but if one occurs, so MUST the other
duration / repeat /

; the following is OPTIONAL,
; but MUST NOT occur more than once
attach
)

disppropext = *

; the following are REQUIRED,
; but MUST NOT occur more than once
description /

; 'duration' and 'repeat' are both OPTIONAL,
; and MUST NOT occur more than once each,
; but if one occurs, so MUST the other
duration / repeat
)

emailpropext = *

; the following are all REQUIRED,
; but MUST NOT occur more than once
description / summary /

; the following is REQUIRED,
; and MAY occur more than once
attendee /

; 'duration' and 'repeat' are both OPTIONAL,
; and MUST NOT occur more than once each,
; but if one occurs, so MUST the other

duration / repeat

)

4. Alarm Unique Identifier

This extension adds a "UID" property to "VALARM" components to allow a unique identifier to specified. The value of this property can then be used to refer uniquely to the "VALARM" component.

The "UID" property defined here follows the definition in Section 3.8.4.7 of [RFC5545] with the security and privacy updates in Section 5.3 of [RFC7986]. In particular it MUST be a globally unique identifier that does not contain any security- or privacy-sensitive information.

The "VALARM" component defined in Section 3 is extended here as:

alarmprop /= *(  

; the following is OPTIONAL,
; but MUST NOT occur more than once

uid

)

5. Alarm Acknowledgement

There is currently no way for a "VALARM" component to indicate whether it has been triggered and acknowledged. With the advent of a standard client/server protocol for calendaring and scheduling data ([RFC4791]) it is quite possible for an event with an alarm to exist on multiple clients in addition to the server. If each of those is responsible for performing the action when an alarm triggers, then multiple "alerts" are generated by different devices. In such a situation, a calendar user would like to be able to "dismiss" the alarm on one device and have it automatically dismissed on the others too.
Also, with recurring events that have alarms, it is important to know when the last alarm in the recurring set was acknowledged, so that the client can determine whether past alarms have been missed.

To address these needs, this specification adds an "ACKNOWLEDGED" property to "VALARM" components to indicate when the alarm was last sent or acknowledged. This is defined by the syntax below.

```
alarmprop /= *(
    ; the following is OPTIONAL,
    ; but MUST NOT occur more than once
    acknowledged
)
```

5.1. Acknowledged Property

Property Name: ACKNOWLEDGED

Purpose: This property specifies the UTC date and time at which the corresponding alarm was last sent or acknowledged.

Value Type: DATE-TIME

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

Conformance: This property can be specified within "VALARM" calendar components.

Description: This property is used to specify when an alarm was last sent or acknowledged. This allows clients to determine when a pending alarm has been acknowledged by a calendar user so that any alerts can be dismissed across multiple devices. It also allows clients to track repeating alarms or alarms on recurring events or to-dos to ensure that the right number of missed alarms can be tracked.

Clients SHOULD set this property to the current date-time value in UTC when a calendar user acknowledges a pending alarm. Certain kinds of alarm may not provide feedback as to when the calendar user sees them, for example email based alerts. For those kinds of alarms, the client SHOULD set this property when the alarm is triggered and the action successfully carried out.
When an alarm is triggered on a client, clients can check to see if an "ACKNOWLEDGED" property is present. If it is, and the value of that property is greater than or equal to the computed trigger time for the alarm, then the client SHOULD NOT trigger the alarm. Similarly, if an alarm has been triggered and an "alert" presented to a calendar user, clients can monitor the iCalendar data to determine whether an "ACKNOWLEDGED" property is added or changed in the alarm component. If the value of any "ACKNOWLEDGED" property in the alarm changes and is greater than or equal to the trigger time of the alarm, then clients SHOULD dismiss or cancel any "alert" presented to the calendar user.

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

acknowledged = "ACKNOWLEDGED" acknowledgedparam ":" datetime CRLF

acknowledgedparam = *( ; the following is OPTIONAL, ; and MAY occur more than once
";" other-param )

Example: The following is an example of this property:

ACKNOWLEDGED:20090604T084500Z

6. Snoozing Alarms

Users often want to "snooze" an alarm, and this specification defines a standard approach to accomplish that.

To "snooze" an alarm, clients create a new "VALARM" component within the parent component of the "VALARM" that was triggered and is being "snoozed" (i.e., as a "sibling" component of the "VALARM" being snoozed). The new "VALARM" MUST be set to trigger at the user’s chosen "snooze" interval after the original alarm triggered. Clients SHOULD use an absolute "TRIGGER" property with a "DATE-TIME" value specified in UTC.

When the "snooze" alarm is triggered and dismissed the client SHOULD remove the corresponding "VALARM" component, or set the "ACKNOWLEDGED" property (see Section 5.1). Alternatively, if the "snooze" alarm is itself "snoozed", the client SHOULD remove the
original "snooze" alarm and create a new one, with the appropriate trigger time and relationship set.

7. Alarm Proximity Trigger

VALARMs are currently triggered when a specific date-time is reached. It is also desirable to be able to trigger alarms based on location, e.g. when arriving at or departing from a particular location.

This specification adds the following properties to "VALARM" components to indicate when an alarm can be triggered based on location.

"PROXIMITY" - indicates that a location based trigger is to be used and which direction of motion is used for the trigger

"STRUCTURED-LOCATION" [I-D.ietf-calext-eventpub-extensions] - used to indicate the actual location to trigger off, specified using a geo: URI [RFC5870] which allows for two or three coordinate values with an optional uncertainty

```plaintext
alarmprop /= *(  
    ; the following is OPTIONAL,  
    ; but MUST NOT occur more than once  
    proximity /  
    ; the following is OPTIONAL,  
    ; and MAY occur more than once, but only  
    ; when a PROXIMITY property is also present  
    structured-location  
)
```

Typically, when a "PROXIMITY" property is used there is no need to specify a time-based trigger using the "TRIGGER" property. However, since "TRIGGER" is defined as a required property for a "VALARM" component, for backwards compatibility it has to be present, but ignored. To indicate a "TRIGGER" that is to be ignored, clients SHOULD use a value a long time in the past. A value of "19760401T005545Z" has been commonly used for this purpose.
7.1.  Proximity Property

Property Name: PROXIMITY

Purpose: This property indicates that a location based trigger is applied to an alarm.

Value Type: TEXT

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

Conformance: This property can be specified within "VALARM" calendar components.

Description: This property is used to indicate that an alarm has a location-based trigger. Its value identifies the direction of motion used to trigger the alarm. One or more location values are set using "STRUCTURED-LOCATION" properties.

When the property value is set to "ARRIVE", the alarm is triggered when the calendar user agent arrives in the vicinity of any of the specified locations. When set to "DEPART", the alarm is triggered when the calendar user agent departs from the vicinity of any specified locations.

When the property value is set to "CONNECT", the alarm is triggered when the calendar user agent connects to a Bluetooth(R) [BTcore]-enabled automobile. When set to "DISCONNECT", the alarm is triggered when the calendar user agent disconnects from a Bluetooth(R)-enabled automobile.

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

proximity = "PROXIMITY" proximityparam ":" proximityvalue CRLF
proximityparam = *
    ; the following is OPTIONAL, ; and MAY occur more than once
    (";" other-param)
)
proximityvalue = "ARRIVE" / "DEPART" / "CONNECT" / "DISCONNECT" / iana-token / x-name

Daboo & Murchison       Expires September 6, 2019               [Page 9]
Example: The following is an example of this property:

PROXIMITY:ARRIVE

7.2. Example

The following example shows a "VALARM" component with a proximity trigger set to trigger when the device running the calendar user agent leaves the vicinity defined by the structured location property. Note use of the "u=" parameter with the "geo" URI to define the precision of the location determination.

BEGIN:VALARM
UID:77D80D14-906B-4257-963F-85B1E734DBB6
TRIGGER;VALUE=DATE-TIME:19760401T005545Z
ACTION:DISPLAY
DESCRIPTION:Remember to buy milk
TRIGGER;VALUE=DATE-TIME:19760401T005545Z
PROXIMITY:DEPART
STRUCTURED-LOCATION;VALUE=URI:geo:40.443,-79.945;u=10
END:VALARM

8. Security Considerations

VALARMS, if not monitored properly, can be used to "spam" users and/or leak personal information. For instance, an unwanted audio or display alert could be considered spam. Or an email alert could be used to leak a user’s location to a third party or to send unsolicited email to multiple users. Therefore, CalDAV clients and servers that accept iCalendar data from a third party (e.g. via iTIP [RFC5546], a subscription feed, or a shared calendar) SHOULD remove all VALARMS from the data prior to storing in their calendar system.

9. IANA Considerations

9.1. Property Registrations

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGED</td>
<td>Current</td>
<td>RFCXXXX, Section 5.1</td>
</tr>
<tr>
<td>PROXIMITY</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
</tbody>
</table>
9.2. Proximity Value Registry

This document creates a new iCalendar registry for values of the "PROXIMITY" property:

+------------+---------+----------------------+
<table>
<thead>
<tr>
<th>Value</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRIVE</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>DEPART</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
</tbody>
</table>
+------------+---------+----------------------+

10. Acknowledgments

This specification came about via discussions at the Calendaring and Scheduling Consortium. Also, thanks to the following for providing feedback: Bernard Desruisseaux, Mike Douglass, Jacob Farkas, Jeffrey Harris, and Ciny Joy.

11. References

11.1. Normative References

[D-I-D.ietf-calext-eventpub-extensions]


11.2. Informative References


11.3. URIs


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

Changes in -05:

1. Added Murchison as editor.
2. Updated keywords boilerplate.
3. Added reference to UID security/privacy recommendations.
4. Removed default alarms.
5. Removed ALARM-AGENT property.
6. Added text about using TRIGGER value in the past in addition to ACTION:NONE to have a default alarm be ignored.
7. Removed text about related alarms.
8. Removed URL alarm action.

9. Added reference to draft-ietf-calext-eventpub-extensions for STRUCTURED-LOCATION.

10. Added CONNECT and DISCONNECT PROXIMITY property values.

11. Added Security Considerations.


Changes in -04:
1. Changed "ID" to "AGENT-ID".
2. Add more text on using "ACKNOWLEDGED" property.
3. Add "RELATED-TO" as a valid property for VALARMs.
4. Add "SNOOZE" relationship type for use with VALARMs.
5. State that "TRIGGER" is typically ignored in proximity alarms.
6. Added "PROXIMITY" value registry.
7. Added a lot more detail on default alarms including new action and property.

Changes in -03: none - resubmission of -02

Changes in -02:
1. Updated to 5545 reference.
2. Clarified use of absolute trigger in UTC in snooze alarms
3. Snooze alarms should be removed when completed
4. Removed status and replaced last-triggered by acknowledged property
5. Added location-based trigger
6. IANA registry tables added

Changes in -01:
1. Removed DESCRIPTION as an allowed property in the URI alarm.
2. Added statement about what to do when ALARM-AGENT is not present.

3. Allow multiple ALARM-AGENT properties to be present.

4. Removed SNOOZE-UNTIL - snoozing now accomplished by creating a new VALARM.

5. Remove VALARM by reference section.

6. Added more detail to CalDAV default alarms.

Authors’ Addresses

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

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

Kenneth Murchison (editor)
FastMail US LLC
1429 Walnut St, Suite 1201
Philadelphia, PA  19102
USA

Email: murch@fastmailteam.com
URI:   http://www.fastmail.com/
This document adds headers to control and restrict the scheduling behaviour of CalDAV servers when updating calendaring resources.
1. Introduction


[RFC6638] defines the "Schedule-Reply" header in Section 8.1, however this header is not sufficient for controlling scheduling in all cases.

Cases where it might be necessary to update the data store on a server without causing scheduling messages to be sent include backup after a data loss event on the server, or importing calendar events from another system.

Calendar server operators deal with these other needs by either using a different method than CalDAV to update their server, or by adding a custom method to suppress scheduling. This document defines a standard method to suppress scheduling, allowing CalDAV to be directly used for restores and imports.

Complex sites can have users who have multiple aliases, and in the most complex cases, a user may have multiple identities who are present on a scheduling event as organizer and/or attendee. When an event is updated over CalDAV, the server must calculate or guess which of those addresses the current user is acting as. This document defines a header which allows the client to inform the server precisely which address they are acting as when adding, modifying or removing a resource.
2. Conventions Used In This Document

In examples, "C:" indicates data sent by a client that is connected to a server. "S:" indicates data sent by the server to the client.

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] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

3. Extending the CalDAV OPTIONS response

A server supporting the features described in this document MUST include "scheduling-controls" as a field in the DAV response header from an OPTIONS request. A value of "scheduling-controls" in the DAV response header indicates to clients that the server supports all the requirements specified in this document.

3.1. Example: Using OPTIONS for the Discovery of Scheduling Controls Support

Request:

OPTIONS /home/brong/calendars/ HTTP/1.1
Host: cal.example.com

Response:

HTTP/1.1 200 OK
Allow: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, COPY, MOVE
Allow: PROPFIND, PROPPATCH, LOCK, UNLOCK, REPORT, ACL
DAV: 1, 2, 3, access-control, calendar-access, scheduling-controls
Date: Thu, 8 Feb 2019 10:16:37 GMT
Content-Length: 0

4. New headers

This document adds two new headers for use on PUT, PROPPATCH and DELETE:

4.1. Scheduling header

Scheduling: {all|none|internal-only|external-only|X-...}

Default: all
Not providing this header, or providing the value of "all", instructs the server to follow the behaviour in [RFC6638] Section 3.2.

Providing the value "none" instructs the server to perform no scheduling at all, and to just store the event (useful for restoring from backup).

The value "internal-only" instructs the server to update the events in other calendars within its system where that can be done silently, but not to send visible notifications to users (where permitted by policy). This is useful when importing multiple related calendars into a new system without flooding external parties with notifications.

The value "external-only" instructs the server to import the data without updating local calendars, but to send notifications to external attendees so they are aware of the event. This is useful when migrating calendar events to a new system where external parties need to have a way to update their participation status in the new system.

  e.g.

  Scheduling: none

TODO: specify error codes

4.2. Schedule-User-Address header

  Schedule-User-Address: URI

  Default: not present

  If this header is not present, the server will calculate the address from the authenticated user, or from the CALDAV:schedule-user-address property on the calendar or principal.

  If this header is provided, it overrides the server’s internal calculation, and informs the server to perform any scheduling as the specified user.

  TODO: specify error codes

  e.g.

  Schedule-User-Address: mailto:foo@example.com
5. Implementation considerations

Any server implementing this extension MUST ensure it has a way to validate Schedule-User-Address settings.

6. IANA Considerations

TODO: IANA request for OPTIONS item

TODO: IANA request for named headers

7. Security Considerations

The "Scheduling" header only allows reduction of the cases in which the server will creating scheduling requests. This is generally good for user privacy, allowing copies of events to be updated without notifying the owner or attendees. This is particularly valuable for cleaning up spam.

The "Schedule-User-Address" header allows the client to override the server choice of address for the user to act as. Servers MUST ensure that the authenticated user has permission to act as the specified address, as well as applying any local policy limitations.

8. Acknowledgments

- Lucia Kristiansen, Google
- CalConnect
- The calext working group

9. Version History

Remove before publishing

9.1. v01, 2019-03-08

- correct name in acknowledgements

9.2. v00, 2019-02-08

- Initial draft based on discussion at CalConnect about Google and FastMail private implementations.
10. Normative References


Author’s Address

Bron Gondwana (editor)
FastMail
Level 2, 114 William St
Melbourne VIC 3000
Australia

Email: brong@fastmailteam.com
URI:     https://www.fastmail.com
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 noncompliance with multiple best current practices of HTTP.
publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. introduction ................................................. 3
   1.1. Rationale for Informational Status .................. 4
2. Conventions Used in This Document .......................... 4
3. Overview ................................................... 4
   3.1. Requirements .......................................... 5
   3.2. Discovering Support for Managed Attachments ............ 5
   3.3. POST Request for Managing Attachments .................. 5
      3.3.1. action= Query Parameter ........................... 6
      3.3.2. rid= Query Parameter ............................. 6
      3.3.3. managed-id= Query Parameter ....................... 7
   3.4. Adding attachments .................................... 7
   3.5. Updating Attachments .................................. 10
   3.6. Removing Attachments via POST .......................... 13
   3.7. Adding Existing Managed Attachments via PUT ............. 15
   3.8. Updating Attachments via PUT .......................... 15
   3.9. Removing Attachments via PUT .......................... 15
   3.10. Retrieving Attachments ................................. 16
   3.11. Error Handling ........................................ 16
   3.12. Additional Considerations ............................ 17
      3.12.1. Quotas ........................................... 17
      3.12.2. Access Control .................................... 17
      3.12.3. Redirects ......................................... 18
      3.12.4. Processing Time ................................... 18
      3.12.5. Automatic Clean-Up by Servers ..................... 18
      3.12.6. Sending Scheduling Messages with Attachments ...... 18
      3.12.7. Migrating Calendar Data ........................... 18
4. Modifications to iCalendar Syntax ............................ 19
   4.1. SIZE Property Parameter ............................... 19
   4.2. FILENAME Property Parameter ............................ 19
   4.3. MANAGED-ID Property Parameter .......................... 20
5. Additional Message Header Fields ............................. 20
   5.1. Cal-Managed-ID Response Header Field ................... 20
6. Additional WebDAV Properties ................................. 21
   6.1. CALDAV:managed-attachments-server-URL property .......... 21
   6.2. CALDAV:max-attachment-size property .................... 22
   6.3. CALDAV:max-attachments-per-resource property ........... 23
7. Implementation Status ....................................... 23
8. Security Considerations ..................................... 25
9. IANA Considerations ......................................... 26
1. Introduction

The iCalendar [RFC5545] data format is used to represent calendar data and is used with iCalendar Transport-independent Interoperability Protocol (iTIP) [RFC5546] to handle scheduling operations between calendar users.

[RFC4791] defines the Calendaring Extensions to WebDAV (CalDAV), 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.
1.1. Rationale for Informational Status

Although this extension to CalDAV has wide deployment, its design does not comply with some of the best current practices of HTTP, namely:

- All operations on attachments are modeled as HTTP POST operations, where the actual type of operation is specified using a query parameter, instead of using separate HTTP POST, PUT, and DELETE methods where appropriate.

- Specific query strings are hardwired into the protocol in violation of Section 2.4 of [RFC7320].

Additionally, this extension misuses the Content-Disposition header field [RFC6266] as a request header field to convey a filename for an attachment rather than using an existing request header field suitable for that purpose, such as "Slug" (see Section 9.7 of [RFC5023]).

Rather than creating interoperability problems with deployed code by updating the design of this extension to be compliant with best current practices and to allow this specification to be placed on the Standards Track, it was decided to simply document how existing implementations interoperate and to publish the document as Informational.

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 BCP 14 [1] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 token in the DAV response header field (as defined in Section 10.1 of [RFC4918]) 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 also include "calendar-managed-attachments-no-recurrence" as a token 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. These requests are subject to the preconditions listed in
Section 3.11. 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:

https://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. For example, to target a recurrence defined by property RECURRENCE-ID;TZID=America/Montreal:20111022T160000, the query parameter rid=20111022T160000 would be used.
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 (targeting the master instance and a specific overridden instance):

https://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" (Section 4.3) 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:

https://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. Note that the use of Content-Disposition as a request header field is nonstandard and specific to this protocol.

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>
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://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>
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.
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)

If a managed attachment is used in more than calendar resource, servers SHOULD NOT change either the "MANAGED-ID" parameter value or the "ATTACH" property value for these attachments - this ensures that clients do not have to download the attachment data again if they already have it cached. Additionally, servers SHOULD validate "SIZE" parameter values and replace incorrect values with the actual sizes of existing attachments.

These PUT requests are subject to the preconditions listed in Section 3.11.

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-action): The action query parameter in the POST request MUST contain one of "attachment-add", "attachment-update", or "attachment-remove".

(CALDAV:valid-rid): The rid query parameter in the POST request MUST NOT be present for an attachment-update action, and MUST contain the value "M" and/or values corresponding to "RECURRENCE-ID" property values in the iCalendar data targeted by the request.

(CALDAV:valid-managed-id): The managed-id query parameter in the POST request MUST NOT be present for an attachment-add action, and 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])
A PUT request to add or modify an existing calendar object resource can make reference to an existing managed attachment. The following new preconditions are 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).

If a precondition for a request is not satisfied:

1. The response status of the request MUST either be 403 (Forbidden), if the request should not be repeated because it will always fail, or 409 (Conflict), if it is expected that the user might be able to resolve the conflict and resubmit the request.

2. The appropriate XML element MUST be returned as the child of a top-level DAV:error element in the response body.

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 a 307 (Temporary Redirect) [RFC7231] or 308 (Permanent Redirect) [RFC7538] response to require the client to re-issue the POST request using a different request-URI. As a result, clients SHOULD 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
```

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:https://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:
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 within the scope of the CalDAV server. This property parameter MUST NOT be present in the case of non-managed attachments.

Example:

ATTACH;MANAGED-ID=aUNhbGVuZGFy:https://attachments.example.com/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.

ABNF:

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

Example:

Cal-Managed-ID:aUNhbGVuZGFy

The Cal-Managed-ID header field MUST only be sent by an origin server in response to a successful POST request with an action set to attachment-add or attachment-update. It MUST only appear once in a response and MUST NOT appear in trailers.
The Cal-Managed-ID header field is end to end and MUST be forwarded by intermediaries. Intermediaries MUST NOT insert, delete, or modify a Cal-Managed-ID header field.

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, the reference to [RFC7942], and any resulting "URIs" references sub-section. >
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 [2] 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 [3].

7.2. Cyrus Server

The open source Cyrus Server [4] 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 [5].

7.3. Oracle Communications Calendar Server

The Oracle Communications Calendar Server [6] 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.
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 [7] client is a standards-compliant client implementing the CalDAV protocol. This production level implementation supports all the requirements described in this document and successfully interoperares 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 [8] 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 [9] 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 [10].

7.7. 2Do

2Do [11] is a standards-compliant 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

The security considerations in [RFC4791] and [RFC4918] apply to this extension. Additionally, servers need to be aware that a client could attack underlying storage by POSTing extremely large
attachments and could attack processing time by uploading a recurring event with a large number of overrides and then repeatedly adding, updating, and deleting attachments.

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]:

<table>
<thead>
<tr>
<th>Property Parameter</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE</td>
<td>Current</td>
<td>RFCXXXX, Section 4.1</td>
</tr>
<tr>
<td>FILENAME</td>
<td>Current</td>
<td>RFCXXXX, Section 4.2</td>
</tr>
<tr>
<td>MANAGED-ID</td>
<td>Current</td>
<td>RFCXXXX, Section 4.3</td>
</tr>
</tbody>
</table>

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


11.2. Informative References


11.3. URIs


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

Changes in calext-04:

1. Added text explaining why this document is being published as Informational.
2. Further specified Cal-Managed-ID per Section 8.3.1 of RFC 7231.
3. Specified that the MANAGED-ID parameter value is unique within the scope of the server.
4. Added more text regarding preconditions.
5. Added text about specific DoS attack vectors.
6. Editorial changes from Gren Elliot and Phillip Kewisch.
7. Editorial changes from Adam Roach.
8. Editorial changes from Alexey Melnikov.
10. Minor editorial changes.

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".
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]).
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
HTTP/1.1 412 Precondition Failed
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://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@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@example.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>
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://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@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@example.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 an overridden instance is created with the RECURRENCE-ID property value matching the value of the "rid" query parameter in the request. Also note that the server takes significant time to complete the request and notifies the client accordingly.

HTTP/1.1 100 Continue

HTTP/1.1 102 Processing

HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://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@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@example.com
ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
FILENAME=agenda.html:https://cal.example.com/attach/65/34XZ2R
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
END:VEVENT

ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@example.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 US LLC
1429 Walnut St, Suite 1201
Philadelphia, PA 19102
USA

Email: murch@fastmailteam.com
URI:   http://www.fastmail.com/
Event Publishing Extensions to iCalendar
draft-ietf-calext-eventpub-extensions-13

Abstract

This specification updates RFC5545 by introducing 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 November 27, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
   1.1. Conventions Used in This Document ...................... 3
2. Components and properties .................................... 3
3. Typed References ................................................ 4
   3.1. Use Cases .................................................. 5
       3.1.1. Piano Concert Performance ............................ 5
       3.1.2. Itineraries ........................................... 5
           3.1.2.1. Reserving facilities ............................ 5
4. Modifications to Calendar Components ............................ 6
5. New Property Parameters ........................................ 7
   5.1. Loctype .................................................... 7
   5.2. Restype .................................................... 7
   5.3. Order ..................................................... 8
   5.4. Schema .................................................... 9
   5.5. Derived ................................................... 9
6. Redefined Property SOURCE ....................................... 10
7. New Properties .................................................. 12
   7.1. Participant Type ........................................... 12
   7.2. Calendar Address .......................................... 14
   7.3. Styled-Description ........................................ 15
   7.4. Structured-Location ....................................... 17
   7.5. Structured-Resource ....................................... 19
   7.6. Structured-Data ............................................ 20
8. New Components .................................................. 23
   8.1. Participant ................................................ 23
   8.2. Schedulable Participant ................................... 25
9. Extended examples ................................................ 26
   9.1. Example 1 .................................................. 26
   9.2. Example 2 .................................................. 26
10. Security Considerations .......................................... 27
11. Privacy Considerations .......................................... 27
12. IANA Considerations ............................................ 28
   12.1. Additional iCalendar Registrations ........................ 28
       12.1.1. Properties ........................................... 28
       12.1.2. Parameters ........................................... 28
       12.1.3. Components ........................................... 28
       12.2. New Registration Tables ................................ 29
           12.2.1. Participant Types ................................. 29
           12.2.2. Resource Types .................................... 29
13. Acknowledgements ............................................... 29
14. References ...................................................... 30
   14.1. Normative References .................................... 30
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 non-standard properties as defined in [RFC5545] section 3.8.8.2.

This document updates [RFC5545] to define 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 [RFC2426] are likely to be most useful to the receivers of such events as they may be used in other applications - such as address books.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 defines PARTICIPANT as a component rather than a property. This is a better match for the way [W3C.REC-xml-20081126] and JSON [RFC8259] handle 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.

Many people or groups may participate in an event. The PARTICIPANT component provides such detailed information. Participants may act as attendees to the event (or derived events) or may just provide a reference – perhaps for mailing lists.

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 presenting 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, address, region, country, postal code as well as other informations such as the 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 can also be used to provide useful extra data about an attendee. For example a LOCATION property inside the
PARTICIPANT gives the actual location of a remote attendee. (But see the note about privacy.)

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 [RFC2426] 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, car hire companies and the hotel.

3.1.2.1. Reserving facilities

For a meeting, the size of a room and the equipment needed depends to some extent on the number of attendees actually in the room.

A meeting may have 10 attendees non of which are co-located. The current ATTENDEE property does not allow for the addition of such
meta-data. The PARTICIPANT property allows attendees to specify their location.

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.
    ;
    styledescription / strucloc / strucres / sdataprop
    ;
    )

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

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

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

jourprop =/ *(
    ;
    ; The following are OPTIONAL,
    ; and MAY occur more than once.
    ;
    styledescription / sdataprop
    ;
    )
freebusyc = "BEGIN" ":" "VFREEBUSY" CRLF
fbprop *participantc
"END" ":" "VFREEBUSY" CRLF

fbprop =/ *{
 ;
 ; The following are OPTIONAL,
 ; and MAY occur more than once.
 ;
 ; stylesheetdescription
 ;
 ;}

5. New Property Parameters

This specification makes use of the LABEL 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" "=" restypevalue CRLF

restypevalue = ("ROOM"
/ "PROJECTOR"
/ "REMOTE-CONFERENCE-AUDIO"
/ "REMOTE-CONFERENCE-VIDEO"
/ iana-token) ; Other IANA-registered
; values

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

The registered values are described below. New resource types SHOULD be registered in the manner laid down in this specification.

ROOM: A room for the event/meeting.

PROJECTOR: Projection equipment.

REMOTE-CONFERENCE-AUDIO: Audio remote conferencing facilities.

REMOTE-CONFERENCE-VIDEO: Video remote conferencing facilities.

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 specifies 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.
When any ORDER parameters have the same value all the associated properties appear as a group within which there is no defined 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.

This parameter MUST NOT be applied to a property that does not allow multiple instances.

Example uses: The ORDER may be applied to the PARTICIPANT-TYPE property to indicate the relative importance of the participant, for example as a sponsor or a performer. For example, ORDER=1 could define the principal performer or soloist.

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
```

5.5. Derived

Parameter Name: DERIVED

Purpose: To specify that the value of the associated property is derived from some other property value or values.
Format Definition:

This parameter is defined by the following notation:

```
derivedparam    = "DERIVED" "=" ("TRUE" / "FALSE")
```
; Default is FALSE

Description: This property parameter can be specified on any property when the value is derived from some other property or properties. When present with a value of TRUE clients MUST NOT update the property.

As an example, if a STYLED-DESCRIPTION property is present with FMTTYPE="application/rtf" then there may be an additional STYLED-DESCRIPTION property with FMTTYPE="text/html" and DERIVED=TRUE and a value created from the rtf value.

Example:

```
STYLED-DESCRIPTION;FMTTYPE=text/html;
DERIVED=TRUE:<html>...
```

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. For example, that information may be a vcard or a plain text typed value.

For value type URI and embedded in a VEVENT or VTODO it may provide a location from which the component may be refreshed.

Value type: There is no default value type for this property. It may be set to URI as in [RFC7986]. 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 can be specified once in an iCalendar object.

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

```plaintext
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 below.

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

Conformance: This property MUST be specified once 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 property is defined by the following notation:

```
participanttype   = "PARTICIPANT-TYPE" partvalueparam ":" partvalue CRLF
partvalue         = ("ACTIVE"
                        / "INACTIVE"
                        / "SPONSOR"
                        / "CONTACT"
                        / "BOOKING-CONTACT"
                        / "EMERGENCY-CONTACT"
                        / "PUBLICITY-CONTACT"
                        / "PLANNER-CONTACT"
                        / "PERFORMER"
                        / "SPEAKER"
                        / iana-token) ; Other IANA-registered values
partvalueparam    = *(; the following is OPTIONAL
                     ; and MAY occur more than once
                     ; (":" other-param)
                     )
```

Example:

The following is an example of this property:

```
PARTICIPANT-TYPE:SPEAKER
```

The registered values for the PARTICIPANT-TYPE property have the meanings described here:

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

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 once 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 property is defined by the following notation:

    calendaraddress   = "CALENDAR-ADDRESS" caladdressparam "::" cal-address CRLF

    caladdressparam   = *( ; the following is OPTIONAL ; and MAY occur more than once ; (*;" other-param)

    )
7.3. Styled-Description

Property name: STYLED-DESCRIPTION

Purpose: This property provides for one or more rich-text descriptions to replace 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, derived and language property parameters can be specified on this property.

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

If it does appear more than once there MUST be exactly one instance of the property with no DERIVED parameter or DERIVED=FALSE. All others MUST have DERIVED=TRUE.

Additionally, if there is one or more STYLED-DESCRIPTION property then the DESCRIPTION property should be either absent or have the parameter DERIVED=TRUE.

Description: This property supports rich-text descriptions, for example HTML. Event publishers typically wish to provide more and better formatted information about the event.

This property is used in the "VEVENT" and "VTTODO" 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. In the PARTICIPANT component it provides a detailed description of the participant.

VALUE=TEXT is used to provide rich-text inline as the property value.

VALUE=URI is used to provide a link to rich-text content which is expected to be displayed inline as part of the event.
In either case the DESCRIPTION property should be absent or contain a plain text rendering of the styled text.

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" and process the content as defined in [W3C.REC-html51-20171003]

Multiple STYLED-DESCRIPTION properties may be used to provide different formats or different language variants. However all but one MUST have DERIVED=TRUE.

Format Definition:

This property is defined by the following notation:

styledescription = "STYLED-DESCRIPTION" styledescparam ":" styledescval CRLF

styledescparam = *(  
; The following is REQUIRED, 
; but MUST NOT occur more than once.  
; (";" "VALUE" "=" ("URI" / "TEXT")) /  
; The following are OPTIONAL,  
; but MUST NOT occur more than once.  
; (";" altrepparam) / (";" languageparam) /  
; (";" fmttypeparam) / (";" derivedparam) /  
; the following is OPTIONAL  
; and MAY occur more than once  
; (";" other-param)  
)

styledescval = ( uri / text )  
;Value MUST match value type

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

```plaintext
STYLE-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, related 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: There may be a number of locations associated with an event. This provides detailed information about these locations.

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 SHOULD match that of the content and of the LANGUAGE parameter if present.

Use of the related parameter: This allows a location to define the start and/or end timezone of the associated component. If a location is specified with a RELATED parameter then the affected DTSTART or DTEND properties MUST be specified as floating DATE-TIME value.

If the RELATED parameter is present with a value of START, then the "DTSTART" property MUST be present in the associated "VEVENT" or "VTTODO" calendar component.

For an event, if the RELATED parameter is present with a value of END, then the "DTEND" property or the "DTSTART" and "DURATION" properties MUST be present in the associated "VEVENT" calendar component.
For a to-do with a RELATED value of END, then either the "DUE" property or the "DTSTART" and "DURATION " properties MUST be present in the associated "VTODO" calendar component.

If there is a location specified with RELATED=START and no location is specified with RELATED=END then the event is assumed to start and end in the same timezone.

Format Definition:

This property is defined by the following notation:

\[
\text{strucloc} = \text{"STRUCTURED-LOCATION" struclocparam \"\":\" struclocval CRLF}
\]

\[
\text{struclocparam} = *\{
; The following is REQUIRED,
; but MUST NOT occur more than once.
; 
; (\"; "VALUE\" =\" (\"URI\" / \"TEXT\") \/) 
; 
; ; the following are OPTIONAL
; ; but MUST NOT occur more than once
; 
; (\"; fmttypeparam) /
; (\"; labelparam) /
; (\"; languageparam) /
; (\"; trigrelparam) /
; (\"; loctypeparam) /
; 
; ; the following is OPTIONAL
; ; and MAY occur more than once
; 
; (\"; other-param) 
\}
\]

\[
\text{struclocval} = ( \text{uri / text} ) 
\]

;Value MUST match value type

Example:

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

\[
\text{STRUCTURED-LOCATION;LABEL="The venue";}
\]

\[
\text{VALUE=URI:}
\]

\[
\text{http://dir.example.com/venues/big-hall.vcf}
\]
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. Typically a resource is anything that might be required or used by a calendar entity and possibly has a directory entry.

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: The property can be specified multiple times in the "VEVENT" or "VTTODO" calendar components.

Description: When used in a component the value of this property provides information about resources used for the event such as rooms, projectors, conferencing capabilities.

Such resources may be a room or a projector. This RESTYPE value registry provides a place in which resource types may be registered for use by scheduling services.

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 ":"
strucresval CRLF
```

```
strucresparam = *

; The following is REQUIRED,
; but MUST NOT occur more than once.

(";" "VALUE" "=" ("URI" / "TEXT")) / 

; 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)
```

```
strucresval = ( uri / text )

;Value MUST match value type
```

Example:

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

```
STRUCTURED-RESOURCE;value=uri;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: 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.

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"
    ) /
    ( 
        ";" "VALUE" ":" "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:{
  "@context": "http://schema.org",
  "@type": "SportsEvent",
  "homeTeam": "Pittsburgh Pirates",
  "awayTeam": "San Francisco Giants"
}
```
8. New Components

8.1. Participant

Component name: PARTICIPANT

Purpose: This component provides information about a participant in an event or task.

Conformance: This component can be specified multiple times in a "VEVENT", "VTODO", "VJOURNAL", or "VFREEBUSY" calendar component.

Description: This component provides information about a 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 CALENDAR-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 component is defined by the following notation:

```
participantc  = "BEGIN" ":" "PARTICIPANT" CRLF
partprop      = *(
  ;
  ; The following are REQUIRED,
  ; but MUST NOT occur more than once.
  ;
  dtstamp / participanttype / uid /
  ;
  ; The following are OPTIONAL,
  ; but MUST NOT occur more than once.
  ;
  created / description / geo / last-mod / priority / seq /
  source / status / calendaraddress / summary / url /
  ;
  ; The following are OPTIONAL,
  ; and MAY occur more than once.
  ;
  attach / categories / comment /
  contact / location / rstatus / related /
  resources / strucloc / strucres / styleddescription /
  iana-prop
  ;
)
```

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

Privacy Issues: When a LOCATION is supplied it provides information
about the location of a participant at a given time or times.
This may represent an unacceptable privacy risk for some
participants. User agents MUST NOT include this information
without informing the participant.

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: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: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. 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.

9.1. 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
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";VALUE=URI:
    http://dir.example.com/venues/big-hall.vcf
STRUCTURED-LOCATION;LABEL="Parking for the venue";VALUE=URI:
    http://dir.example.com/venues/parking.vcf
IMAGE;VALUE=URI;DISPLAY=BADGE;FMTTYPE=image/png:
    http://example.com/images/concert.png
BEGIN:PARTICIPANT
    SOURCE:http://example.com/sponsor.vcf
END:PARTICIPANT
BEGIN:PARTICIPANT
    SOURCE:http://www.example.com/people/johndoe.vcf
END:PARTICIPANT
END:VEVENT
```
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 planning
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

10. 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.

When processing HTML content applications need to be aware of the many security and privacy issues as described in the IANA considerations section of [W3C.REC-html51-20171003]

11. 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].

The addition of location information to the new participant component provides information about the location of participants at a given time.
12. IANA Considerations

12.1. Additional iCalendar Registrations

12.1.1. Properties

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALENDAR-ADDRESS</td>
<td>Current</td>
<td>RFCXXX, Section 7.2</td>
</tr>
<tr>
<td>PARTICIPANT-TYPE</td>
<td>Current</td>
<td>RFCXXX, Section 7.1</td>
</tr>
<tr>
<td>SOURCE</td>
<td>Current</td>
<td>RFCXXX, Section 6</td>
</tr>
<tr>
<td>STRUCTURED-DATA</td>
<td>Current</td>
<td>RFCXXX, Section 7.6</td>
</tr>
<tr>
<td>STYLED-DESCRIPTION</td>
<td>Current</td>
<td>RFCXXX, Section 7.3</td>
</tr>
<tr>
<td>STRUCTURED-LOCATION</td>
<td>Current</td>
<td>RFCXXX, Section 7.4</td>
</tr>
<tr>
<td>STRUCTURED-RESOURCE</td>
<td>Current</td>
<td>RFCXXX, Section 7.5</td>
</tr>
</tbody>
</table>

12.1.2. Parameters

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

<table>
<thead>
<tr>
<th>Property Parameter</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCTYPE</td>
<td>Current</td>
<td>RFCXXX, Section 5.1</td>
</tr>
<tr>
<td>ORDER</td>
<td>Current</td>
<td>RFCXXX, Section 5.3</td>
</tr>
<tr>
<td>RESTYPE</td>
<td>Current</td>
<td>RFCXXX, Section 5.2</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Current</td>
<td>RFCXXX, Section 5.4</td>
</tr>
</tbody>
</table>

12.1.3. Components

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

<table>
<thead>
<tr>
<th>Component</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTICIPANT</td>
<td>Current</td>
<td>RFCXXX, Section 8.1</td>
</tr>
</tbody>
</table>
12.2. New Registration Tables

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

12.2.1. Participant Types

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

<table>
<thead>
<tr>
<th>Participant Type</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>INACTIVE</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>SPONSOR</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>CONTACT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>BOOKING-CONTACT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>EMERGENCY-CONTACT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>PUBLICITY-CONTACT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>PLANNER-CONTACT</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>PERFORMER</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
<tr>
<td>SPEAKER</td>
<td>Current</td>
<td>RFCXXXX, Section 7.1</td>
</tr>
</tbody>
</table>

12.2.2. Resource Types

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

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECTOR</td>
<td>Current</td>
<td>RFCXXXX, Section 5.2</td>
</tr>
<tr>
<td>ROOM</td>
<td>Current</td>
<td>RFCXXXX, Section 5.2</td>
</tr>
<tr>
<td>REMOTE-CONFERENCE-AUDIO</td>
<td>Current</td>
<td>RFCXXXX, Section 5.2</td>
</tr>
<tr>
<td>REMOTE-CONFERENCE-VIDEO</td>
<td>Current</td>
<td>RFCXXXX, Section 5.2</td>
</tr>
</tbody>
</table>

13. 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.

14. References

14.1. Normative References


14.2. Informative References

[iana-property-registry]
"IANA iCalendar Element Registries",
<https://www.iana.org/assignments/icalendar/icalendar.xhtml>.

Appendix A. Open issues

None at the moment

Appendix B. Change log

calext-v13 2019-05-26 MD
  o Respond to various issues.

calext-v12 2019-02-28 MD
  o Fix styled-description example. Respond to various AD issues.
    Some typos.

calext-v11 2019-02-27 MD
  o Add DERIVED parameter for styled-description, RELATED parameter for
    structured-location

calext-v09 2018-08-30 MD
  o Sorted out inconsistencies in refs to 5546
calext-v08 2018-07-06 MD

- Add some text for equal ORDER values
- Switched scheduleaddress to calendaraddress in participant abnf. Also added more properties
- Fixed PARTICIPANT abnf

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/PARTICIPANT-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
o STYLED-DESCRIPTION to handle rich text

o Much more...

2011-01-07

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

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

o 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
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. In contrast to the JSON-based jCal format, it is not a direct mapping from iCalendar and expands semantics where appropriate.

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 December 26, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
Table of Contents

1.  Introduction .................................................. 4
   1.1.  Relation to the iCalendar format ...................... 4
   1.2.  Relation to the jCal format ............................ 5
   1.3.  Notational Conventions .................................. 5
2.  JSCalendar objects ........................................... 5
   2.1.  JSEvent .................................................. 5
   2.2.  JSTask ................................................... 6
   2.3.  JSGroup ................................................ 6
3.  Structure of JSCalendar objects ............................... 6
   3.1.  Type signatures .......................................... 6
   3.2.  Data Types ................................................ 7
      3.2.1.  UTCDate .............................................. 7
      3.2.2.  LocalDate ............................................ 7
      3.2.3.  Duration ............................................. 7
      3.2.4.  PatchObject ......................................... 8
      3.2.5.  Identifiers .......................................... 9
      3.2.6.  Time Zones .......................................... 9
      3.2.7.  Normalization and equivalence ........................ 9
   3.3.  Custom property extensions and values .................. 10
4.  Common JSCalendar properties ................................ 10
   4.1.  Metadata properties ..................................... 10
      4.1.1.  @type ................................................. 10
      4.1.2.  uid .................................................. 11
      4.1.3.  relatedTo ........................................... 11
      4.1.4.  prodId ............................................... 12
      4.1.5.  created ............................................. 12
      4.1.6.  updated ............................................. 12
      4.1.7.  sequence ............................................ 12
      4.1.8.  method ............................................... 13
   4.2.  What and where properties ................................ 13
      4.2.1.  title ............................................... 13
      4.2.2.  description .......................................... 13
      4.2.3.  descriptionContentType ............................ 13
      4.2.4.  showWithoutTime ..................................... 13
      4.2.5.  locations ........................................... 14
      4.2.6.  virtualLocations ..................................... 15
      4.2.7.  links ................................................ 15
      4.2.8.  locale ............................................... 16
      4.2.9.  keywords ............................................ 17
      4.2.10. categories ........................................... 17
      4.2.11. color ................................................ 17
   4.3.  Recurrence properties ................................... 17
      4.3.1.  recurrenceRule ....................................... 17
4.3.2. recurrenceOverrides ........................................... 23
4.3.3. excluded ....................................................... 24
4.4. Sharing and scheduling properties ................................ 24
4.4.1. priority ......................................................... 24
4.4.2. freeBusyStatus ................................................ 25
4.4.3. privacy ......................................................... 25
4.4.4. replyTo ......................................................... 26
4.4.5. participants ................................................... 27
4.5. Alerts properties .................................................. 30
4.5.1. useDefaultAlerts .............................................. 30
4.5.2. alerts .......................................................... 30
4.6. Multilingual properties .......................................... 32
4.6.1. localizations ................................................... 32
4.7. Time zone properties ............................................ 33
4.7.1. timeZone ........................................................ 33
5. Type-specific JSCalendar properties ................................ 35
5.1. JSEvent properties ................................................. 35
5.1.1. start .......................................................... 35
5.1.2. timeZone ....................................................... 35
5.1.3. duration ....................................................... 35
5.1.4. status ........................................................ 36
5.2. JSTask properties .................................................. 36
5.2.1. due ............................................................ 36
5.2.2. start .......................................................... 36
5.2.3. timeZone ....................................................... 36
5.2.4. estimatedDuration ............................................. 36
5.2.5. statusUpdatedAt ............................................... 37
5.2.6. progress ....................................................... 37
5.2.7. status ........................................................ 38
5.3. JSGroup properties ............................................... 38
5.3.1. entries ........................................................ 39
5.3.2. source ........................................................ 39
6. JSCalendar object examples ...................................... 39
6.1. Simple event ..................................................... 39
6.2. Simple task ....................................................... 40
6.3. Simple group ..................................................... 40
6.4. All-day event ...................................................... 41
6.5. Task with a due date ............................................. 41
6.6. Event with end time-zone ....................................... 42
6.7. Floating-time event (with recurrence) .......................... 42
6.8. Event with multiple locations and localization ............... 43
6.9. Recurring event with overrides ................................ 44
6.10. Recurring event with participants ................................ 45
7. Security Considerations ............................................ 47
8. IANA Considerations ................................................ 47
9. Acknowledgments ................................................... 48
10. References .......................................................... 48
10.1. Normative References ........................................... 48
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] and [RFC7986] and extensions, 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 is mostly a pragmatic choice: its widespread use makes JSCalendar easier to adopt, and the ready availability of production-ready JSON implementations eliminates a whole category of parser-related 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 have a way to express a concise difference between two calendar components, which results in verbose exchanges during scheduling.

1.2. Relation to the jCal format

The JSON format for iCalendar data, jCal [RFC7265], is a direct mapping between iCalendar and JSON. It does not attempt to extend or update iCalendar semantics, and consequently does not address the issues outlined in Section 1.1.

Since the standardization of jCal, the majority of implementations and service providers either kept using iCalendar, or came up with their own proprietary JSON representation, which often are incompatible with each other. JSCalendar is intended to meet this demand for JSON formatted calendar data, and to provide a standard representation as an alternative to new proprietary formats.

1.3. 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/jscalendar+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/jscalendar+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/jscalendar+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. Properties are specified as being either mandatory or optional. Optional properties may have a default value, if explicitly specified in the property definition.

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.
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, including fractional seconds. For example "2006-01-02T15:04:05" and "2006-01-02T15:04:05.003" are both valid. 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 time zone at the same wall-clock time (as opposed to the same instant point in time).

3.2.3. Duration

A Duration object 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"
dur-week    = 1*DIGIT "W"
duration    = "P" (dur-day [dur-time] / dur-time / dur-week)

In addition, the duration MUST NOT include fractional second values unless the fraction is non-zero.

A SignedDuration object is represented as a duration, optionally preceded by a sign character. It typically is used to express the offset of a point in time relative to an associated time. It is specified by the following ABNF:

signed-duration = (["+" ] / "- ") duration

A negative sign indicates a point in time at or before the associated time, a positive or no sign a time at or after the associated time.

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/foo/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. Identifiers

If not stated otherwise in the respective property definition, properties and object keys that define identifiers MUST be string values, MUST be at least 1 character and maximum 256 characters in size, and MUST only contain characters from the "URL and Filename safe" Base 64 Alphabet, as defined in section 5 of [RFC4648]. This is the ASCII alphanumeric characters (A-Za-z0-9), hyphen (-), and underscore (_). Note that [RFC7493] requires string values be encoded in UTF-8, so the maximum size of an identifier according to this definition is 256 octets.

Identifiers in object maps need not be universally unique, e.g. two calendar objects MAY use the same identifiers in their respective "links" properties.

Nevertheless, a UUID typically is a good choice.

3.2.6. Time Zones

By default, time zones in JSCalendar are identified by their name in the IANA Time Zone Database [1], and the zone rules of the respective zone record apply.

Implementations MAY embed the definition of custom time zones in the "timeZones" property (see Section 4.7.1).

3.2.7. Normalization and equivalence

JSCalendar aims to provide unambiguous definitions for value types and properties, but does not define a general normalization 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).
Normalization 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 normalization 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 normalization 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 5 describe the set of supported properties per type.

4.1. Metadata properties

4.1.1. @type

Type: String (mandatory).

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).
4.1.2. uid

Type: String (mandatory).

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.

For compatibility with [RFC5545] UIDs, implementations MUST be able to receive and persist values of at least 255 octets for this property, but they MUST NOT truncate values in the middle of a UTF-8 multi-octet sequence.

4.1.3. relatedTo

Type: String[Relation] (optional).

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[Boolean] (optional). Describes how the linked object is related to this object as a set of relation types. If not null, the set MUST NOT be empty.

  Keys in the set MUST be one of the following values, defined in a future specification 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.

  The value for each key in the set MUST be "true".
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 "next" relation MUST be set on the original object’s relatedTo property for the UID of the new object. A "first" relation for 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 (optional).

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]. It MUST only use characters of an iCalendar TEXT data value (see section 3.3.11 in [RFC5545]).

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 (optional).

The date and time this object was initially created.

4.1.6. updated

Type: UTCDate (mandatory).

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

4.1.7. sequence

Type: Number (optional, default: "0").

 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 (optional).

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

4.2. What and where properties

4.2.1. title

Type: String (optional, default: empty String).

A short summary of the object.

4.2.2. description

Type: String (optional, default: empty String).

A longer-form text description of the object. The content is formatted according to the "descriptionContentType" property.

4.2.3. descriptionContentType

Type: String (optional, default: "text/plain").

Describes the media type ([RFC6838]) of the contents of the "description" property. Media types MUST be sub-types of type "text", and SHOULD be "text/plain" or "text/html" ([MIME]). They MAY define parameters and the "charset" parameter value MUST be "utf-8", if specified. Descriptions of type "text/html" MAY contain "cid" URLs ([RFC2392]) to reference links in the calendar object by use of the "cid" property of the Link object.

4.2.4. showWithoutTime

Type: Boolean (optional, default: "false").

Indicates the time is not important to display to the user when rendering this calendar object, for example an event that conceptually occurs all day or across multiple days, such as "New Year's Day" or "Italy Vacation". While the time component is important for free-busy calculations and checking for scheduling clashes, calendars may choose to omit displaying it and/or display the object separately to other objects to enhance the user's view of their schedule.
4.2.5. locations

Type: String[Location] (optional).

A map of location identifiers to Location objects, representing locations associated with the object.

A Location object has the following properties. It must define at least one other property than the "relativeTo" property.

- name: String (optional, default: empty String). The human-readable name of the location.
- description: String (optional). Human-readable, plain-text instructions for accessing this location. This may be an address, set of directions, door access code, etc.
- relativeTo: String (optional). 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 if this property is omitted.

* "start": The JSCalendar object starts at this location.
* "end": The JSCalendar object ends at this location.
- timeZone: String (optional). A time zone for this location. Also see Section 3.2.6.
- linkIds: String[Boolean] (optional). A set of link ids for links to alternate representations of this location. Each key in the set MUST be the identifier of a Link object defined in the "links" property of this calendar object. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

For example, an alternative representation could be in vCard format.
4.2.6. virtualLocations

Type: String[VirtualLocation] (optional).

A map of identifiers to VirtualLocation objects, representing virtual locations, such as video conferences or chat rooms, associated with the object.

A VirtualLocation object has the following properties.

- name: String (optional, default: empty String). The human-readable name of the virtual location.
- description: String (optional). Human-readable plain-text instructions for accessing this location. This may be an address, set of directions, door access code, etc.
- uri: String (mandatory). A URI that represents how to connect to this virtual 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.

4.2.7. links

Type: String[Link] (optional).

A map of link identifiers to Link objects, representing external resources associated with the object.

A Link object has the following properties:

- href: String (mandatory). 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 arbitrarily large data in JSCalendar object instances.

- cid: String (optional). This MUST be a valid "content-id" value according to the definition of section 2 in [RFC2392]. The identifier MUST be unique within this JSCalendar object Link objects but has no meaning beyond that. Specifically, it MAY be different from the link identifier in the enclosing "links" property.
type: String (optional). The content-type [RFC6838] of the resource, if known.

size: Number (optional). 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). Identifies the relation of the linked resource to the object. If set, 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.

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 "display" property MAY be set to indicate the purpose of this image.

display: String (optional). Describes the intended purpose of a link to an image. If set, the "rel" property MUST be set to "icon". 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


4.2.8. locale

Type: String (optional).

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

    Type: String[Boolean] (optional).

    A set of keywords or tags that relate to the object. The set is
    represented as a map, with the keys being the keywords. The value
    for each key in the map MUST be "true".

4.2.10.  categories

    Type: String[Boolean] (optional).

    A set of categories that relate to the calendar object. The set is
    represented as a map, with the keys being the categories specified as
    URIs. The value for each key in the map MUST be "true".

    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".

4.2.11.  color

    Type: String (optional).

    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 (optional).

    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]. A JSEvent recurs by applying
    the recurrence rule to the start date-time. A JSTask recurs by
    applying the recurrence rule to the start date-time, if defined,
    otherwise it recurs by the due date-time, if defined. If the task
    neither defines a start or due date-time, its "recurrenceRule"
    property value MUST be "null".

    A Recurrence object has the following properties:
frequency: String (mandatory). 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, default: "1"). The INTERVAL part from iCalendar. If included, it MUST be an integer "x >= 1".

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

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

firstDayOfWeek: String (optional, default: "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 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, 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.
A recurrence rule specifies a set of set of date-times for recurring calendar objects. A recurrence rule has the following semantics. Note, wherever "year", "month" or "day of month" is used, this is within the calendar system given by the "rscale" property, which defaults to gregorian if omitted.

1. A set of candidates is generated. This is every second within a period defined by the frequency property value:
"yearly": every second from midnight on the 1st day of a year (inclusive) to midnight the 1st day of the following year (exclusive).

If skip is not "omit", the calendar system has leap months and there is a byMonth property, generate candidates for the leap months even if they don’t occur in this year.

If skip is not "omit" and there is a byMonthDay property, presume each month has the maximum number of days any month may have in this calendar system when generating candidates, even if it’s more than this month actually has.

"monthly": every second from midnight on the 1st day of a month (inclusive) to midnight on the 1st of the following month (exclusive).

If skip is not "omit" and there is a byMonthDay property, presume the month has the maximum number of days any month may have in this calendar system when generating candidates, even if it’s more than this month actually has.

"weekly": every second from midnight (inclusive) on the first day of the week (as defined by the firstDayOfWeek property, or Monday if omitted), to midnight 7 days later (exclusive).

"daily": every second from midnight at the start of the day (inclusive) to midnight at the end of the day (exclusive).

"hourly": every second from the beginning of the hour (inclusive) to the beginning of the next hour (exclusive).

"minutely": every second from the beginning of the minute (inclusive) to the beginning of the next minute (exclusive).

"secondly": the second itself, only.

2. Each date-time candidate is compared against all of the byX properties of the rule except bySetPosition. If any property in the rule does not match the date-time, it is eliminated. Each byX property is an array; the date-time matches the property if it matches any of the values in the array. The properties have the following semantics:

byMonth: the date-time is in the given month.

byWeekNo: the date-time is in the nth week of the year. Negative numbers mean the nth last week of the year. This
corresponds to weeks according to week numbering as defined in ISO.8601.2004, with a week defined as a seven day period, starting on the firstDayOfWeek property value or Monday if omitted. Week number one of the calendar year is the first week that contains at least four days in that calendar year.

If the date-time is not valid (this may happen when generating candidates with a skip property in effect), it is always eliminated by this property.

* byYearDay: the date-time is on the nth day of year. Negative numbers mean the nth last day of the year.

If the date-time is not valid (this may happen when generating candidates with a skip property in effect), it is always eliminated by this property.

* byMonthDay: the date-time is on the given day of the month. Negative numbers mean the nth last day of the month.

* byDay: the date-time is on the given day of the week. If the day is prefixed by a number, it is the nth occurrence of that day of the week within the month (if frequency is monthly) or year (if frequency is yearly). Negative numbers means nth last occurrence within that period.

* byHour: the date-time has the given hour value.

* byMinute: the date-time has the given minute value.

* bySecond: the date-time has the given second value.

If a skip property is defined and is not "omit", there may be candidates that do not correspond to valid dates (e.g. 31st February in the gregorian calendar). In this case, the properties MUST be considered in the order above and:

1. After applying the byMonth filter, if the candidate’s month is invalid for the given year increment it (if skip is "forward") or decrement it (if skip is "backward") until a valid month is found, incrementing/decrementing the year as well if you pass through the beginning/end of the year. This only applies to calendar systems with leap months.

2. After applying the byMonthDay filter, if the day of the month is invalid for the given month and year, change the date to the first day of the next month (if skip == "forward") or the last day of the current month (if skip == "backward").
3. If any valid date produced after applying the skip is already a candidate, eliminate the duplicate. (For example after adjusting, 30th February and 31st February would both become the same "real" date, so one is eliminated as a duplicate.)

3. If a bySetPosition property is included, this is now applied to the ordered list of remaining dates (this property specifies the indexes of date-times to keep; all others should be eliminated. Negative numbers are indexes from the end of the list, with -1 being the last item).

4. Any date-times before the start date of the event are eliminated (see below for why this might be needed).

5. If a skip property is included and is not "omit", eliminate any date-times that have already been produced by previous iterations of the algorithm. (This is not possible if skip == "omit".)

6. If further dates are required (we have not reached the until date, or count limit) skip the next (interval - 1) sets of candidates, then continue from step 1.

When determining the set of occurrence dates for an event or task, the following extra rules must be applied:

1. The start date-time is always the first occurrence in the expansion (and is counted if the recurrence is limited by a "count" property), even if it would normally not match the rule.

2. The first set of candidates to consider is that which would contain the start date-time. This means the first set may include candidates before the start; such candidates are eliminated from the results in step (4) as outlined before.

3. The following properties MUST be implicitly added to the rule under the given conditions:

   * If frequency > "secondly" and no bySecond property: Add a bySecond property with the sole value being the seconds value of the start date-time.

   * If frequency > "minutely" and no byMinute property: Add a byMinute property with the sole value being the minutes value of the start date-time.

   * If frequency > "hourly" and no byHour property: Add a byHour property with the sole value being the hours value of the start date-time.
* If frequency is "weekly" and no byDay property: Add a byDay property with the sole value being the day-of-the-week of the start date-time.

* If frequency is "monthly" and no byDay property and no byMonthDay property: Add a byMonthDay property with the sole value being the day-of-the-month of the start date-time.

* If frequency is "yearly" and no byYearDay property:
  + if there are no byMonth or byWeekNo properties, and either there is a byMonthDay property or there is no byDay property: Add a byMonth property with the sole value being the month of the start date-time.
  + if there is no byMonthDay, byWeekNo or byDay properties: Add a byMonthDay property with the sole value being the day-of-the-month of the start date-time.
  + if there is a byWeekNo property and no byMonthDay or byDay properties: Add a byDay property with the sole value being the day-of-the-week of the start date-time.

4.3.2. recurrenceOverrides

Type: LocalDate[PatchObject] (optional).

A map of the recurrence-ids (the date-time of the start of the occurrence) to 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.

If the patch object defines the "excluded" property value to be "true", then the recurring calendar object does not occur at the recurrence-id date-time (like an EXDATE from iCalendar). Such a patch object MUST NOT patch any other property.

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 of the LocalDate key. However, individual properties of the occurrence can be modified by a patch, or multiple patches. It is valid to patch the start property value, and this patch takes precedence over the LocalDate key. Both the LocalDate key as well as
the patched start date-time may occur before the original JSCalendar object’s start or due date.

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

- @type
- o  @type
- o  uid
- o  relatedTo
- o  prodId
- o  method
- o  recurrenceRule
- o  recurrenceOverrides
- o  replyTo

4.3.3. excluded

Type: Boolean (optional, default: "false").

Defines if this object is an overridden, excluded instance of a recurring JSCalendar object (also see Section 4.3.2). If this property value is "true", this calendar object instance MUST be removed from the occurrence expansion. The absence of this property or its default value "false" indicates that this instance MUST be added to the occurrence expansion.

4.4. Sharing and scheduling properties

4.4.1. priority

Type: Number (optional, default: "0").

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 (optional, default: "busy").

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 (optional, default: "public").

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. The following properties MAY be
  shared, any other properties MUST NOT be shared:

* @type
* created
* due
* duration
* estimatedDuration
* freeBusyStatus
* privacy
* recurrenceOverrides. Only patches whose keys are prefixed with one of the above properties are allowed to be shared.
* sequence
* showWithoutTime
* start
* timeZone
* timeZones
* uid
* updated

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

4.4.4. replyTo

Type: String[String] (optional).

Represents methods by which participants may submit their RSVP response to the organizer of the calendar object. The keys in the property value are the available methods and MUST only contain ASCII alphanumeric characters (A-Za-z0-9). 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, but MUST preserve the method key and URI. This property MUST be omitted if no method is defined (rather than an empty object). If this property is set, the "participants" property of this calendar object MUST contain at least one participant.

The following methods are defined:

  o "imip": The organizer accepts an iMIP [RFC6047] response at this email address. The value MUST be a "mailto:" URI.
o  "web": Opening this URI in a web browser will provide the user with a page where they can submit a reply to the organizer.

o  "other": The organizer is identified by this URI but the method how to submit the RSVP is undefined.

4.4.5. participants

Type: String[Participant] (optional).

A map of participant identifiers to participants, describing their participation in the calendar object.

If this property is set, then the "replyTo" property of this calendar object MUST define at least one reply method.

A Participant object has the following properties:

o  name: String (optional). The display name of the participant (e.g. "Joe Bloggs").

o  email: String (optional). The email address for the participant.

o  sendTo: String[String]. Represents methods by which the participant may receive the invitation and updates to the calendar object.

The keys in the property value are the available methods and MUST only contain ASCII alphanumeric characters (A-Za-z0-9). 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, but MUST preserve the method key and URI. This property MUST be omitted if no method is defined (rather than an empty object).

The following methods are defined:

*  "imip": The participant accepts an iMIP [RFC6047] request at this email address. The value MUST be a "mailto:" URI. It MAY be different from the value of the participant’s "email" property.

*  "other": The participant is identified by this URI but the method how to submit the invitation or update is undefined.

o  kind: String (optional). What kind of entity this participant is, if known.
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 if this property is omitted.

* "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.

o roles: String[Boolean]. A set of roles that this participant fulfills.

At least one role MUST be specified for the participant. The keys in the set MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:

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

The value for each key in the set MUST be "true". Roles that are unknown to the implementation MUST be preserved and MAY be ignored.

o locationId: String (optional). 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 omitted.

o participationStatus: String (optional, default: "needs-action"). The participation status, 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.

- **attendance**: String (optional, default: "required"). The required attendance 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".

  - "none": Indicates a participant who is copied for information purposes only.
  - "optional": Indicates a participant whose attendance is optional.
  - "required": Indicates a participant whose attendance is required.

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

- **scheduleSequence**: Number (optional, default: "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 participant 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 (optional). The "updated" property of the last iMIP response from the participant.

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

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

- **delegatedTo**: String[Boolean] (optional). A set of participant ids that this participant has delegated their participation to. Each
key in the set MUST be the identifier of a participant. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

- delegatedFrom: String[Boolean] (optional). A set of participant ids that this participant is acting as a delegate for. Each key in the set MUST be the identifier of a participant. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

- memberOf: String[Boolean] (optional). A set 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). Each key in the set MUST be the identifier of a participant. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

- linkIds: String[Boolean] (optional). A set of links to more information about this participant, for example in vCard format. The keys in the set MUST be the identifier of a Link object in the calendar object’s "links" property. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

4.5. Alerts properties

4.5.1. useDefaultAlerts

Type: Boolean (optional, default: "false").

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] (optional).

A map of alert identifiers to Alert objects, representing alerts/reminders to display or send the user for this calendar object.

An Alert Object has the following properties:

- trigger: OffsetTrigger|UnknownTrigger. Defines when to trigger the alert.
An *OffsetTrigger* object has the following properties:

* type: String (mandatory). The value of this property MUST be "offset".

* offset: SignedDuration (mandatory). Defines to trigger the alert relative to the time property defined in the "relativeTo" property. 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.

* relativeTo: String (optional, default: "start"). Specifies the time property which the alert offset is relative to. The value MUST be one of:
  + "start": triggers the alert relative to the start of the calendar object
  + "end": triggers the alert relative to the end/due time of the calendar object

An *UnknownTrigger* object is an object that contains a *type* property whose value is not "offset", plus zero or more other properties. This is for compatibility with client extensions and future RFCs. Implementations MUST NOT trigger for trigger types they do not understand, but MUST preserve them.

- acknowledged: UTCDate (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 the "recurrenceOverrides" property.

- snoozed: UTCDate (optional).

  If the user temporarily dismisses the alert, this is the UTC date-time after which it should trigger again. Setting this property on an instance of a recurring calendar object MUST update the alarm on the top-level object, unless the respective instance already is defined in "recurrenceOverrides". It MUST NOT generate an override for the sole use of snoozing an alarm.
4.6. Multilingual properties

4.6.1. localizations

Type: String[PatchObject] (optional).

A map of [RFC5646] language tags to patch objects, which localize 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. In addition to all the restrictions on patches specified there, the pointer also MUST NOT start with one of the following prefixes; any patch with a such a key MUST be ignored:

- @type
- due
- duration
- freeBusyStatus
- localization
- method
- participants
- prodId
- progress
- relatedTo
Note that this specification does not define how to maintain validity of localized content. For example, a client application changing a JSCalendar object’s title property might also need to update any localizations of this property. Client implementations SHOULD provide the means to manage localizations, but how to achieve this is specific to the application’s workflow and requirements.

4.7. Time zone properties

4.7.1. timeZones

Type: String[TimeZone] (optional).

Maps identifiers of custom time zones to their time zone definition. The following restrictions apply for each key in the map:

- It MUST start with the "/" character (ASCII decimal 47; also see sections 3.2.19 of [RFC5545] and 3.6. of [RFC7808] for discussion of the forward slash character in time zone identifiers).

- It MUST be a valid "paramtext" value as specified in section 3.1. of [RFC5545].

- At least one other property in the same JSCalendar object MUST reference a time zone using this identifier (i.e. orphaned time zones are not allowed).

An identifier need only be unique to this JSCalendar object.

A TimeZone object maps a VTIMEZONE component from iCalendar ([RFC5545]). A valid time zone MUST define at least one transition rule in the "standard" or "daylight" property. Its properties are:

- tzId: String (mandatory). The TZID property from iCalendar.
- `lastModified`: UTCDate (optional). The `LAST-MODIFIED` property from iCalendar.
- `url`: String (optional). The `TZURL` property from iCalendar.
- `validUntil`: UTCDate (optional). The `TZUNTIL` property from iCalendar specified in [RFC7808].
- `aliases`: String[Boolean] (optional). Maps the `TZID-ALIAS-OF` properties from iCalendar specified in [RFC7808] to a JSON set of aliases. The set is represented as an object, with the keys being the aliases. The value for each key in the set MUST be "true".
- `standard`: TimeZoneRule[] (optional). The `STANDARD` sub-components from iCalendar. The order MUST be preserved during conversion.
- `daylight`: TimeZoneRule[] (optional). The `DAYLIGHT` sub-components from iCalendar. The order MUST be preserved during conversion.

A TimeZoneRule object maps a `STANDARD` or `DAYLIGHT` sub-component from iCalendar, with the restriction that at most one recurrence rule is allowed per rule. It has the following properties:

- `start`: LocalDate (mandatory). The `DTSTART` property from iCalendar.
- `offsetTo`: String (mandatory). The `TZOFFSETTO` property from iCalendar.
- `offsetFrom`: String (mandatory). The `TZOFFSETFROM` property from iCalendar.
- `recurrenceRule`: RecurrenceRule (optional). The `RRULE` property mapped as specified in Section 4.3.1. During recurrence rule evaluation, the "until" property value MUST be interpreted as a local time in the UTC time zone.
- `recurrenceDates`: LocalDate[Boolean] (optional). Maps the `RDATE` properties from iCalendar to a JSON set. The set is represented as an object, with the keys being the recurrence dates. The value for each key in the set MUST be "true".
- `names`: String[Boolean] (optional). Maps the `TZNAME` properties from iCalendar to a JSON set. The set is represented as an object, with the keys being the names. The value for each key in the set MUST be "true".
comments: String[] (optional). Maps the COMMENT properties from iCalendar. The order MUST be preserved during conversion.

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 (mandatory).

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

5.1.2. timeZone

Type: String|null (optional, default: "null").

Identifies 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). Also see Section 3.2.6.

5.1.3. duration

Type: Duration (optional, default: "PT0S").

The zero or positive duration of the event in the event’s start time zone. The same rules as for the iCalendar DURATION value type ([RFC5545]) apply: The duration of a week or a day in hours/minutes/seconds may vary if it overlaps a period of discontinuity in the event’s time zone, for example a change from standard time to daylight-savings time. Leap seconds MUST NOT be considered when computing an exact duration. When computing an exact duration, the greatest order time components MUST be added first, that is, the number of days MUST be added first, followed by the number of hours, number of minutes, and number of seconds. Fractional seconds MUST be added last.

A JSEvent MAY involve start and end locations that are in different time zones (e.g. a trans-continental flight). This can be expressed using the "relativeTo" and "timeZone" properties of the JSEvent’s "location" objects.
5.1.4. status

Type: String (optional, default: "confirmed").

The scheduling status (Section 4.4) of a JSEvent. 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 (optional).

The date/time the task is due in the task’s time zone.

5.2.2. start

Type: LocalDate (optional).

The date/time the task should start in the task’s time zone.

5.2.3. timeZone

Type: String|null (optional, default: "null").

Identifies 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). Also see Section 3.2.6.

5.2.4. estimatedDuration

Type: Duration (optional).

Specifies the estimated positive duration of time the task takes to complete.
5.2.5. statusUpdatedAt

Type: UTCDate (optional).

Specifies the date/time the task status properties was last updated.

If the task is recurring and has future instances, a client may want to keep track of the last status update timestamp of a specific task recurrence, but leave other instances unchanged. One way to achieve this is by overriding the statusUpdatedAt property in the task "recurrenceOverrides" property. However, this could produce a long list of timestamps for regularly recurring tasks. An alternative approach is to split the JSTask into a current, single instance of JSTask with this instance status update time and a future recurring instance. Also see Section 4.1.3 on splitting.

5.2.6. 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 (optional). The progress of the participant for this task, if known. This property MUST NOT be set if the "participationStatus" of this participant is any other value but "accepted".

A ParticipantProgress object has the following properties:

- status: String (mandatory). 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 participant completed their task.
  * "in-process": The participant has started this task.
  * "failed": The participant failed to complete their task.

- timestamp: UTCDate (mandatory). Describes the last time when the participant progress got updated.
5.2.7. status

Type: String (optional).

Defines the overall status of this task. If omitted, the default status (Section 4.4) of a JSTask is defined as follows (in order of evaluation):

- "completed": if the "status" property value of all participant progresses is "completed".
- "failed": if at least one "status" property value of the participant progresses is "failed".
- "in-process": if at least one "status" property value of the participant progresses is "in-process".
- "needs-action": If none of the other criteria match.

If set, it MUST be one of:

- "needs-action": Indicates the task needs action.
- "completed": Indicates the task is completed.
- "in-process": Indicates the task is in process.
- "cancelled": Indicates the task is cancelled.
- "pending": Indicates the task has been created and accepted for processing, but not yet started.
- "failed": Indicates the task failed.

5.3. JSGroup properties

JSGroup supports the following JSCalendar properties (Section 4):

- @type
- uid
- created
- updated
- categories
as well as the following JSGroup-specific properties:

5.3.1. entries

Type: String[JSTask|JSEvent] (mandatory).

A collection of group members. This is represented as a map of the "uid" property value to the JSCalendar object member having that uid. Implementations MUST ignore entries of unknown type.

5.3.2. source

Type: String (optional).

The source from which updated versions of this group may be retrieved from. The value MUST be a URI.

6. 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.

6.1. 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.
6.2. Simple task

This example illustrates a simple task for a plain to-do item.

```json
{
  "@type": "jstask",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
  "updated": "2018-01-15T18:00:00Z",
  "title": "Do something"
}
```

6.3. Simple group

This example illustrates a simple calendar object group that contains an event and a task.

```json
{
  "@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"
}
```
6.4. 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",
  "showWithoutTime": true,
  "start": "1900-04-01T00:00:00",
  "duration": "P1D",
  "recurrenceRule": {
    "frequency": "yearly"
  }
}

6.5. 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.
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. The location names can serve as input for navigation systems.

```
{
  "...": "",
  "title": "Flight XY51 to Tokyo",
  "start": "2018-04-01T09:00:00",
  "timeZone": "Europe/Berlin",
  "duration": "PT10H30M",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "rel": "start",
      "name": "Frankfurt Airport (FRA)"
    },
    "c2c7ac67-dc13-411e-a7d4-0780fb61fb08": {
      "rel": "end",
      "name": "Narita International Airport (NRT)"
    }
  }
}
```

6.7. 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.
6.8. 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.

```json
{
...
"title": "Yoga",
"start": "2018-01-01T07:00:00",
"duration": "PT30M",
"recurrenceRule": {
  "frequency": "daily"
}
}
```
6.9. 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 no course is held. On January 5 at 2pm is an optional introduction course, that occurs before the first regular lecture.
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.

```json
{
  "...": "",
  "title": "FooBar team meeting",
}
```
"start": "2018-01-08T09:00:00",
"timeZone": "Africa/Johannesburg",
"duration": "PT1H",
"virtualLocations": {
  "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
    "name": "ChatMe meeting room",
    "uri": "https://chatme.example.com?id=1234567"
  }
},
"recurrenceRule": {
  "frequency": "weekly"
},
"replyTo": {
  "imip": "mailto:6489-4f14-a57f-c1@schedule.example.com"
},
"participants": {
  "dG9tQGZvb2Jhci5leGItcGxlMmNvbQ": {
    "name": "Tom Tool",
    "email": "tom@foobar.example.com",
    "sendTo": {
      "imip": "mailto:6489-4f14-a57f-c1@calendar.example.com"
    },
    "participationStatus": "accepted",
    "roles": {
      "attendee": true
    }
  },
  "em9lQGZvb2Jhci5leGItcGxlMmNvbQ": {
    "name": "Zoe Zelda",
    "email": "zoe@foobar.example.com",
    "sendTo": {
      "imip": "mailto:zoe@foobar.example.com"
    },
    "participationStatus": "accepted",
    "roles": {
      "owner": true,
      "attendee": true,
      "chair": true
    }
  }
},
"recurrenceOverrides": {
  "2018-03-08T09:00:00": {
    "participants/dG9tQGZvb2Jhci5leGItcGxlMmNvbQ/participationStatus": "declined"
  }
}
7. 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.

Several JSCalendar properties contain URIs as values, and processing these properties requires extra care. Section 7 of [RFC3986] discusses security risk related to URIs.

8. IANA Considerations

This document defines a MIME media type for use with JSCalendar data formatted in JSON.

Type name: application

Subtype name: jscalendar+json

Required parameters: type

The "type" parameter conveys the type of the JSCalendar data in the body part, with the value being one of "jsevent", "jstask", or "jsgroup". The parameter MUST NOT occur more than once. It MUST match the value of the "@type" property of the JSON-formatted JSCalendar object in the body.

Optional parameters: none

Encoding considerations: Same as encoding considerations of application/json as specified in RFC8529, Section 11 [RFC8259].

Security considerations: See Section 7 of this document.

Interoperability considerations: This media type provides an alternative to iCalendar, jCal and proprietary JSON-based calendar data formats.
Published specification: This specification.

Applications that use this media type: Applications that currently make use of the text/calendar and application/calendar+json media types can use this as an alternative. Similarly, applications that use the application/json media type to transfer calendaring data can use this to further specify the content.

Fragment identifier considerations: N/A

Additional information:

- Magic number(s): N/A
- File extensions(s): N/A
- Macintosh file type code(s): N/A

Person & email address to contact for further information:
calext@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: See the "Author’s Address" section of this document.

Change controller: IETF

9. 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.

10. References

10.1. Normative References


10.2. Informative References


10.3. URIs

[1] https://www.iana.org/time-zones

[2] https://www.iana.org/assignments/link-relations/link-relations.xhtml


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