

calext
Internet-Draft
Updates: 6638 (if approved)
Intended status: Standards Track
Expires: September 27, 2019

B. Gondwana, Ed.
FastMail
March 26, 2019

CalDAV Extension for scheduling controls
draft-ietf-calext-caldav-scheduling-controls-00

Abstract

This document adds headers to control and restrict the scheduling behaviour of CalDAV servers when updating calendaring resources.

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 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	2
2. Conventions Used In This Document	3
3. Extending the CalDAV OPTIONS response	3
3.1. Example: Using OPTIONS for the Discovery of Scheduling Controls Support	3
4. New headers	3
4.1. Scheduling header	4
4.2. Schedule-User-Address header	4
5. Implementation considerations	5
6. IANA Considerations	5
7. Security Considerations	5
8. Acknowledgments	5
9. Version History	5
9.1. ietf-calext-v00, 2019-03-26	5
9.2. gondwana-v01, 2019-03-08	6
9.3. gondwana-v00, 2019-02-08	6
10. Normative References	6
Author's Address	6

1. Introduction

[RFC6638] defines automatic scheduling operations for resources stored on [!@RFC4791] CalDAV servers.

[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 create 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

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

9. Version History

Remove before publishing

9.1. ietf-calext-v00, 2019-03-26

- o Adopt into calext working group based on no objections on the list

9.2. gondwana-v01, 2019-03-08

- o correct name in acknowledgements

9.3. gondwana-v00, 2019-02-08

- o Initial draft based on discussion at CalConnect about Google and FastMail private implementations.

10. Normative References

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

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

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>

Network Working Group
Internet-Draft
Updates: 5545,7986 (if approved)
Intended status: Standards Track
Expires: November 27, 2019

M. Douglass
Spherical Cow Group
May 26, 2019

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

14.2. Informative References	31
Appendix A. Open issues	31
Appendix B. Change log	31
Author's Address	34

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.
              ;
              styleddescription / strucloc / strucres / sdataprop
              ;
              )

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

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

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

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

```
freebusyc  = "BEGIN" ":" "VFREEBUSY" CRLF
            fbprop *participantc
            "END" ":" "VFREEBUSY" CRLF

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

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 "FMPTYPE" parameter on the corresponding property.

Example:

```
STRUCTURED-DATA;FMPTYPE=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.

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

```
styleddescription = "STYLED-DESCRIPTION" styleddescparam ":"
                  styleddescval CRLF

styleddescparam   = *(
                    ; 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) /
                    (";" fmimetypeparam) / (";" derivedparam) /
                    ;
                    ; the following is OPTIONAL
                    ; and MAY occur more than once
                    ;
                    (";" other-param)
                    )

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

Example:

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

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

7.4. Structured-Location

Property name: STRUCTURED-LOCATION

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

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

Property Parameters: IANA, non-standard, label, loctype, 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 "VTODO" 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:

```

strucloc      = "STRUCTURED-LOCATION" struclocparam ":"
                struclocval CRLF

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)
)

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

```

Example:

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

```

STRUCTURED-LOCATION;LABEL="The venue";
VALUE=URI:
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 "VTODO" 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)
)

strucewaval   = ( 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"
                ":" binary
              ) /
              (
                ";" "VALUE" "=" "URI"
                ":" uri
              )
              CRLF

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

Example: The following is an example of this property:

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

8. New Components

8.1. Participant

Component name: PARTICIPANT

Purpose: This component provides information about a participant in an event or 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
               "END" ":" "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

- o It contains a CALENDAR-ADDRESS property
- o 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\n
  Piano Sonata No 30
DTSTAMP:20171116T145739Z
DTSTART;TZID=America/New_York:20170315T150000Z
DTEND;TZID=America/New_York:20170315T163000Z
LAST-MODIFIED:20170216T145739Z
SUMMARY:Beethoven Piano Sonatas
UID:123456
STRUCTURED-LOCATION;LABEL="The venue";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;FMPTYPE=image/png:h
  ttp://example.com/images/concert.png
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:SPONSOR
SOURCE:http://example.com/sponsor.vcf
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:PERFORMER:
SOURCE:http://www.example.com/people/johndoe.vcf
END:PARTICIPANT
END:VEVENT
```

9.2. Example 2

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

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

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]:

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

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]:

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

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]:

Component	Status	Reference
PARTICIPANT	Current	RFCXXXX, Section 8.1

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.

Participant Type	Status	Reference
ACTIVE	Current	RFCXXXX, Section 7.1
INACTIVE	Current	RFCXXXX, Section 7.1
SPONSOR	Current	RFCXXXX, Section 7.1
CONTACT	Current	RFCXXXX, Section 7.1
BOOKING-CONTACT	Current	RFCXXXX, Section 7.1
EMERGENCY-CONTACT	Current	RFCXXXX, Section 7.1
PUBLICITY-CONTACT	Current	RFCXXXX, Section 7.1
PLANNER-CONTACT	Current	RFCXXXX, Section 7.1
PERFORMER	Current	RFCXXXX, Section 7.1
SPEAKER	Current	RFCXXXX, Section 7.1

12.2.2. Resource Types

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

Resource Type	Status	Reference
PROJECTOR	Current	RFCXXXX, Section 5.2
ROOM	Current	RFCXXXX, Section 5.2
REMOTE-CONFERENCE-AUDIO	Current	RFCXXXX, Section 5.2
REMOTE-CONFERENCE-VIDEO	Current	RFCXXXX, Section 5.2

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, DOI 10.17487/RFC2426, September 1998, <<https://www.rfc-editor.org/info/rfc2426>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4589] Schulzrinne, H. and H. Tschofenig, "Location Types Registry", RFC 4589, DOI 10.17487/RFC4589, July 2006, <<https://www.rfc-editor.org/info/rfc4589>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.
- [RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <<https://www.rfc-editor.org/info/rfc7986>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.

[W3C.REC-html51-20171003] Faulkner, S., Eicholz, A., Leithead, T., and A. Danilo, "HTML 5.1 2nd Edition", World Wide Web Consortium Recommendation REC-html51-20171003, October 2017, <<https://www.w3.org/TR/2017/REC-html51-20171003>>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <<http://www.w3.org/TR/2008/REC-xml-20081126>>.

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

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

calext-v04 2017-10-11 MD

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

calext-v03 2017-10-09 MD

- o Mostly typographical and other minor changes

calext-v02 2017-04-20 MD

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

calext-v01 2017-02-18 MD

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

calext-v00 2016-08-?? MD

- o Name changed - taken up by calext working group

v06 2016-06-26 MD

- o Fix up abnf

- o change ref to ietf from daboo
 - o take out label spec - use Cyrus spec
- v05 2016-06-14 MD
- o Remove GROUP and HASH. they can be dealt with elsewhere if desired
 - o Change ORDER to integer ≥ 1 .
 - o Incorporate Structured-Data into this specification.
- v04 2014-02-01 MD
- o Added updates attribute.
 - o Minor typos.
 - o Resubmitted mostly to refresh the draft.
- v03 2013-03-06 MD
- o Replace PARTICIPANT with ASSOCIATE plus related changes.
 - o Added section showing modifications to components.
 - o Replace ID with GROUP and modify HASH.
 - o Replace TITLE param with LABEL.
 - o Fixed STYLED-DESCRIPTION in various ways, correct example.
- v02 2012-11-02 MD
- o Collapse sections with description of properties and the use cases into a section with sub-sections.
 - o New section to describe relating properties.
 - o Remove idref and upgrade hash to have the reference
 - o No default value types on properties..
- v01 2012-10-18 MD Many changes.
- o SPONSOR and STRUCTURED-CONTACT are now in PARTICIPANT
 - o 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>

Calendaring extensions
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2019

N. Jenkins
R. Stepanek
FastMail
June 29, 2019

JSCalendar: A JSON representation of calendar data
draft-ietf-calext-jscalendar-17

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 31, 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	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. UTCDateTime	7
3.2.2. LocalDateTime	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.	timeZones	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

10.2. Informative References	51
10.3. URIs	51
Authors' Addresses	51

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:

- o The attributes of the calendar entry represented must be described as a simple key-value pair, reducing complexity of its representation.
- o The data model should avoid all ambiguities and make it difficult to make mistakes during implementation.
- o 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.
- o 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:

- o "Boolean|String": The value is either a JSON "Boolean" value, or a JSON "String" value.

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

3.2. Data Types

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

3.2.1. UTCDateTime

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 offset MUST be the character "Z". 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. LocalDateTime

This is a date-time string with no time zone/offset information. It is otherwise in the same format as UTCDateTime, 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 LocalDateTime 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:

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

- o 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.
- o 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:

- o "jsevent": a JSCalendar event (Section 2.1).

- o "jstask": a JSCalendar task (Section 2.2).
- o "jsgroup": a JSCalendar group (Section 2.3).

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:

- o 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: UTCDateTime (optional).

The date and time this object was initially created.

4.1.6. `updated`

Type: UTCDateTime (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.

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

- o timeZone: String (optional). A time zone for this location. Also see Section 3.2.6.
- o coordinates: String (optional). An [RFC5870] "geo:" URI for the location.
- o 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.

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

- o 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.

- o 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.

- o `type`: String (optional). The content-type [RFC6838] of the resource, if known.
- o `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.
- o `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.

- o `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
- o `title`: String (optional). A human-readable plain-text description of the resource.

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:

- o 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.

- o interval: Number (optional, default: "1"). The INTERVAL part from iCalendar. If included, it MUST be an integer "x >= 1".
- o rscale: String (optional, default: "gregorian"). The RSCALE part from iCalendar RSCALE [RFC7529], converted to lower-case.
- o skip: String (optional, default: "omit"). The SKIP part from iCalendar RSCALE [RFC7529], converted to lower-case.
- o 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".
- o 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.

- o `byMonthDay`: `Number[]` (optional). The BYMONTHDAY part from `iCalendar`. The array MUST have at least one entry if included.
- o `byMonth`: `String[]` (optional). The BYMONTH part from `iCalendar`. Each entry is a string representation of a number, starting from "1" for the first month in the calendar (e.g. "1" means "January" with Gregorian calendar), with an optional "L" suffix (see [RFC7529]) for leap months (this MUST be upper-case, e.g. "3L"). The array MUST have at least one entry if included.
- o `byYearDay`: `Number[]` (optional). The BYYEARDAY part from `iCalendar`. The array MUST have at least one entry if included.
- o `byWeekNo`: `Number[]` (optional). The BYWEEKNO part from `iCalendar`. The array MUST have at least one entry if included.
- o `byHour`: `Number[]` (optional). The BYHOUR part from `iCalendar`. The array MUST have at least one entry if included.
- o `byMinute`: `Number[]` (optional). The BYMINUTE part from `iCalendar`. The array MUST have at least one entry if included.
- o `bySecond`: `Number[]` (optional). The BYSECOND part from `iCalendar`. The array MUST have at least one entry if included.
- o `bySetPosition`: `Number[]` (optional). The BYSETPOS part from `iCalendar`. The array MUST have at least one entry if included.
- o `count`: `Number` (optional). The COUNT part from `iCalendar`. This MUST NOT be included if an "until" property is specified.
- o `until`: `LocalDateTime` (optional). The UNTIL part from `iCalendar`. This MUST NOT be included if a "count" property is specified. Note: if not specified otherwise for a specific JSCalendar object, this date is presumed to be in the time zone specified in "timeZone". As in `iCalendar`, the until value bounds the recurrence rule inclusively.

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: LocalDateTime[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 LocalDateTime 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 LocalDateTime key. Both the

LocalDateTime 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 be ignored if it starts with one of the following prefixes:

- 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:

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

- o "public": The full details of the object are visible to those whom the object's calendar is shared with.
- o "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.
- o 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.
- o expectReply: Boolean (optional, default: "false"). If true, the organizer is expecting the participant to notify them of their status.
- o 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.

- o scheduleUpdated: UTCDateTime (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.

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

- o `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).
- o `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).
- o `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:

- o `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 SHOULD NOT trigger for trigger types they do not understand, but MUST preserve them.

- o *acknowledged*: UTCDateTime (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.

- o *snoozed*: UTCDateTime (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.

- o `action`: String (optional, default: "display"). Describes how to alert the user.

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

- * `"display"`: The alert should be displayed as appropriate for the current device and user context.
- * `"email"`: The alert should trigger an email sent out to the user, notifying about the alert. This action is typically only appropriate for server implementations.

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:

- o `@type`
- o `due`
- o `duration`
- o `freeBusyStatus`
- o `localization`
- o `method`
- o `participants`
- o `prodId`
- o `progress`
- o `relatedTo`

- o sequence
- o start
- o status
- o timeZone
- o uid
- o useDefaultAlerts

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:

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

- o tzId: String (mandatory). The TZID property from iCalendar.

- o `lastModified`: `UTCDateTime` (optional). The `LAST-MODIFIED` property from `iCalendar`.
- o `url`: `String` (optional). The `TZURL` property from `iCalendar`.
- o `validUntil`: `UTCDateTime` (optional). The `TZUNTIL` property from `iCalendar` specified in [RFC7808].
- o `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"`.
- o `standard`: `TimeZoneRule[]` (optional). The `STANDARD` sub-components from `iCalendar`. The order MUST be preserved during conversion.
- o `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:

- o `start`: `LocalDateTime` (mandatory). The `DTSTART` property from `iCalendar`.
- o `offsetTo`: `String` (mandatory). The `TZOFFSETTO` property from `iCalendar`.
- o `offsetFrom`: `String` (mandatory). The `TZOFFSETFROM` property from `iCalendar`.
- o `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.
- o `recurrenceDates`: `LocalDateTime[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"`.
- o `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"`.

- o `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: `LocalDateTime` (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:

- o "confirmed": Indicates the event is definite.
- o "cancelled": Indicates the event is cancelled.
- o "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: LocalDateTime (optional).

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

5.2.2. start

Type: LocalDateTime (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: `UTCDateTime` (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:

- o `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:

- o `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.

- o `timestamp: UTCDateTime` (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):

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

If set, it MUST be one of:

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

5.3. JSGroup properties

JSGroup supports the following JSCalendar properties (Section 4):

- o @type
- o uid
- o created
- o updated
- o categories

- o keywords
- o name
- o description
- o color
- o links

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.

```
{
  "@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"
}
```

6.2. Simple task

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

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

6.3. Simple group

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


```
{
  "@type": "jsgroup",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc343",
  "updated": "2018-01-15T18:00:00Z",
  "name": "A simple group",
  "entries": [
    {
      "@type": "jsevent",
      "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f1",
      "updated": "2018-01-15T18:00:00Z",
      "title": "Some event",
      "start": "2018-01-15T13:00:00",
      "timeZone": "America/New_York",
      "duration": "PT1H"
    },
    {
      "@type": "jstask",
      "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
      "updated": "2018-01-15T18:00:00Z",
      "title": "Do something"
    }
  ]
}
```

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.

```
{
  "...": "",
  "title": "Buy groceries",
  "due": "2018-01-19T18:00:00",
  "timeZone": "Europe/Vienna",
  "estimatedDuration": "PT1H"
}
```

6.6. Event with end time-zone

This example illustrates the use of end time-zones by use of an international flight. The flight starts on April 1, 2018 at 9am in Berlin local time. The duration of the flight is scheduled at 10 hours 30 minutes. The time at the flights destination is in the same time-zone as Tokyo. Calendar clients could use the end time-zone to display the arrival time in Tokyo local time and highlight the time-zone difference of the flight. 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)",
      "timeZone": "Asia/Tokyo"
    }
  }
}
```

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.

```

{
  "...": "",
  "title": "Yoga",
  "start": "2018-01-01T07:00:00",
  "duration": "PT30M",
  "recurrenceRule": {
    "frequency": "daily"
  }
}

```

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 concert on premises or online. The event title and descriptions are localized.

```

{
  "...": "",
  "title": "Live from Music Bowl: The Band",
  "description": "Go see the biggest music event ever!",
  "locale": "en",
  "start": "2018-07-04T17:00:00",
  "timeZone": "America/New_York",
  "duration": "PT3H",
  "locations": {
    "c0503d30-8c50-4372-87b5-7657e8e0fedd": {
      "name": "The Music Bowl",
      "description": "Music Bowl, Central Park, New York",
      "coordinates": "geo:40.7829,73.9654"
    }
  },
  "virtualLocations": {
    "6f3696c6-1e07-47d0-9ce1-f50014b0041a": {
      "name": "Free live Stream from Music Bowl",
      "uri": "https://stream.example.com/the_band_2018"
    }
  },
  "localizations": {
    "de": {
      "title": "Live von der Music Bowl: The Band!",
      "description": "Schau dir das groesste Musikereignis an!",
      "virtualLocations/6f3696c6-1e07-47d0-9ce1-f50014b0041a/name":
        "Gratis Live-Stream aus der Music Bowl"
    }
  }
}

```

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.

```

{
  "...": "",
  "title": "Calculus I",
  "start": "2018-01-08T09:00:00",
  "timeZone": "Europe/London",
  "duration": "PT1H30M",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "title": "Math lab room 1",
      "description": "Math Lab I, Department of Mathematics"
    }
  },
  "recurrenceRule": {
    "frequency": "weekly",
    "until": "2018-06-25T09:00:00"
  },
  "recurrenceOverrides": {
    "2018-01-05T14:00:00": {
      "title": "Introduction to Calculus I (optional)"
    },
    "2018-04-02T09:00:00": {
      "excluded": "true"
    },
    "2018-06-25T09:00:00": {
      "title": "Calculus I Exam",
      "start": "2018-06-25T10:00:00",
      "duration": "PT2H",
      "locations": {
        "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
          "title": "Big Auditorium",
          "description": "Big Auditorium, Other Road"
        }
      }
    }
  }
}

```

6.10. Recurring event with participants

This example illustrates scheduled events. A team meeting occurs every week since January 8, 2018 at 9am Johannesburg time. The event owner also chairs the event. Participants meet in a virtual meeting room. An attendee has accepted the invitation, but on March 8, 2018 he is unavailable and declined participation for this occurrence.

```

{
  "...": "",
  "title": "FooBar team meeting",

```

```
"start": "2018-01-08T09:00:00",
"timeZone": "Africa/Johannesburg",
"duration": "PT1H",
"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": {
  "dG9tQGZvb2Jhci5leGFtcGxlLmNvbQ": {
    "name": "Tom Tool",
    "email": "tom@foobar.example.com",
    "sendTo": {
      "imip": "mailto:6489-4f14-a57f-c1@calendar.example.com"
    },
    "participationStatus": "accepted",
    "roles": {
      "attendee": true
    }
  },
  "em9lQGZvb2Jhci5leGFtcGxlLmNvbQ": {
    "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/dG9tQGZvb2Jhci5leGFtcGxlLmNvbQ/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 calendaring 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

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

- [RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, DOI 10.17487/RFC2392, August 1998, <<https://www.rfc-editor.org/info/rfc2392>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <<https://www.rfc-editor.org/info/rfc4791>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <<https://www.rfc-editor.org/info/rfc5870>>.

- [RFC6047] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 6047, DOI 10.17487/RFC6047, December 2010, <<https://www.rfc-editor.org/info/rfc6047>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <<https://www.rfc-editor.org/info/rfc6901>>.
- [RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", RFC 7265, DOI 10.17487/RFC7265, May 2014, <<https://www.rfc-editor.org/info/rfc7265>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC7529] Daboo, C. and G. Yakushev, "Non-Gregorian Recurrence Rules in the Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 7529, DOI 10.17487/RFC7529, May 2015, <<https://www.rfc-editor.org/info/rfc7529>>.
- [RFC7808] Douglass, M. and C. Daboo, "Time Zone Data Distribution Service", RFC 7808, DOI 10.17487/RFC7808, March 2016, <<https://www.rfc-editor.org/info/rfc7808>>.
- [RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <<https://www.rfc-editor.org/info/rfc7986>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <<https://www.rfc-editor.org/info/rfc8288>>.

10.2. Informative References

[MIME] "IANA Media Types", <<https://www.iana.org/assignments/media-types/media-types.xhtml>>.

10.3. URIs

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

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

[3] <https://www.w3.org/TR/2011/REC-css3-color-20110607/#svg-color>

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>

Calendaring extensions
Internet-Draft
Intended status: Informational
Expires: December 31, 2019

N. Jenkins
R. Stepanek
FastMail
June 29, 2019

JSCalendar: Converting from and to iCalendar
draft-ietf-calext-jscalendar-icalendar-01

Abstract

This document provides an informational guideline for converting JSCalendar from and to iCalendar.

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 31, 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	2
1.1. Motivation	2
1.2. Scope and caveats	3
1.3. Notational Conventions	3
2. New iCalendar parameters	3
2.1. SUBSECOND parameter	3
3. JSEvent	4
4. JSTask	4
5. JSGroup	5
6. Common properties	5
6.1. Time properties and types	8
6.2. Locations	9
6.3. Participants	10
7. Custom properties	12
8. Security Considerations	12
9. IANA Considerations	12
10. Acknowledgments	12
11. References	12
11.1. Normative References	12
11.2. Informative References	13
Authors' Addresses	13

1. Introduction

1.1. Motivation

The JSCalendar [draft-ietf-calex-jscalendar] data format is used to represent calendar data, and is meant as an alternative to the widely deployed iCalendar [RFC5545] data format.

While new calendaring services and applications might use JSCalendar as their main data format to exchange calendaring data, they are likely to interoperate with services and clients that just support iCalendar. Similarly, existing calendaring data is stored in iCalendar format in databases and other calendar stores, and providers and users might want to represent this data also in JSCalendar. Lastly, some implementations might want to preserve custom iCalendar properties, that have no equivalent in JSCalendar when converting between these formats.

To facilitate these use cases, this document provides an informational guide how to convert JSCalendar data from and to iCalendar.

1.2. Scope and caveats

JSCalendar and iCalendar have a lot of semantics in common, but they are not interchangeable formats:

- o JSCalendar contains a richer data model to express calendar information such as event locations and participants; while future iCalendar extensions may allow a direct mapping, for now there may be no representation directly in iCalendar of some properties and these have been marked as implementation specific for mapping.
- o iCalendar may contain arbitrary, non-standardised data with custom properties/attributes. Translating these into JSCalendar is implementation specific.
- o iCalendar has some obsolete features that have been removed from JSCalendar due to not being useful and/or supported in the real world (e.g. custom email alerts to send to random people). Translating these may lose some of the original fidelity.
- o Implementations may use a custom property to store data that could not be mapped directly in either direction in the original or a custom format, however this is not interoperable.

Accordingly, this document does not standardize a canonical translation between iCalendar and JSCalendar, and implementations MUST NOT make any assumptions how iCalendar data is represented in JSCalendar by other systems.

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].

2. New iCalendar parameters

2.1. SUBSECOND parameter

Parameter name: SUBSECOND

Purpose: This parameter is used to define fractional seconds for time values and durations. SUBSECOND MUST NOT be used in date-time calculations or comparisons in iCalendar. It is meant to preserve time precision on time values and duration with sub-second precision, without increasing the time value range within iCalendar.

Description: This parameter MAY be specified on properties of type DATE-TIME or DURATION. The integral part of the float value MUST be zero. The value MUST NOT be negative. iCalendar implementations SHOULD ignore this parameter in date time arithmetic. Implementations MUST ignore presence of the SUBSECOND parameter on RECURRENCE-ID properties when determining recurrence overrides. If present on a RECURRENCE-ID property, its value MUST match the SUBSECOND parameter value on the DATE-TIME property that defines the reference point for the recurring instances.

Format Definition:

This parameter is defined by the following notation:

subsecond-param = float

Example: DTSTART;SUBSECOND=0.03:20190605T133015

3. JSEvent

A JSEvent maps to the the iCalendar VEVENT component type [RFC5545]. The following tables maps the JSEvent-specific properties to iCalendar:

Property	iCalendar counterpart
duration	DURATION property. If the VEVENT contains a DTEND property, the this maps to the duration property as the time span between DTSTART and DTEND when converting the respective time points to the UTC time zone. Fractional seconds SHOULD be preserved with the SUBSECOND parameter.

Table 1: Mapping JSEvent properties

4. JSTask

A JSTask object maps to the iCalendar VTOD0 component type [RFC5545]. The following tables maps the JSTask-specific properties to iCalendar:

Property	iCalendar counterpart
due	Maps to the DUE property. See Section 6.1.
estimatedDuration	ESTIMATED-DURATION property in the RFC draft [draft-apthorp-ical-tasks], or the DURATION property otherwise. Fractional seconds SHOULD be preserved with the SUBSECOND parameter.
statusUpdatedAt	COMPLETED property. The JSTask status property MUST have value "completed". Fractional seconds SHOULD be preserved with the SUBSECOND parameter.
progress	PARTSTAT and COMPLETED properties, including the definitions in the RFC draft [draft-apthorp-ical-tasks].
status	STATUS property, including the definitions in the RFC draft [draft-apthorp-ical-tasks].

Table 2: Mapping JSTask properties

5. JSGroup

A JSGroup maps to a iCalendar VCALENDAR containing VEVENT or VTOD components.

Property	iCalendar counterpart
entries	VEVENT and VTOD components embedded in a VCALENDAR component.
source	SOURCE property.

Table 3: Mapping JSGroup properties

6. Common properties

This section contains recommendations how to map JSCalendar from and to iCalendar. It lists all common JSCalendar object properties in alphabetical order.

Property	iCalendar counterpart
@type	Determined by the iCalendar component type: "jsevent" for VEVENT, "jstask" for VTOD, "jsgroup" for VCALENDAR.
alerts	Each entry maps to a VALARM component. The action property maps to iCalendar ACTION, where both iCalendar "DISPLAY" and "AUDIO" values map to the "display" action. An EMAIL value maps to a JSCalendar "email" action. _relativeTo_ and _offset_ map to the TRIGGER property.
categories	CONCEPT property, defined in [draft-ietf-calext-ical-relations].
color	COLOR property, as specified in [RFC7986].
created	CREATED property. Fractional seconds SHOULD be preserved with the SUBSECOND parameter.
description	DESCRIPTION property.
descriptionContentType	Implementation-specific.
excluded	EXDATE property. Fractional seconds SHOULD be preserved with the SUBSECOND parameter.
freeBusyStatus	TRANSP property.
invitedBy	Implementation-specific.
keywords	CATEGORIES property, as specified in [RFC7986].
links	ATTACH ([RFC5545]), URL or IMAGE ([RFC7986]) properties with URI value types map to the the Link _href_. The FMTTYPE parameter maps to _type_, the SIZE parameter to _size_. Mapping other properties is implementation-specific.

locale	LANGUAGE parameter of the SUMMARY or DESCRIPTION property.
localizations	Implementation-specific.
locations	See Section 6.2.
method	METHOD property of the embedding VCALENDAR.
participants	See Section 6.3.
priority	PRIORITY property.
privacy	CLASS property.
prodId	PRODID property.
recurrenceOverrides	RDATE and EXDATE properties, and any VEVENT or VTODO instances with a recurrence-id and same UID as the mapped main object. If the DTSTART property defines a SUBSECOND parameter, but the RECURRENCE-ID of a recurrence instance does not, then use the SUBSECOND parameter value of DTSTART to determine the recurrence override time stamp.
recurrenceRule	RRULE property. For all-day calendar objects, map the <code>_until_</code> property value to an iCalendar DATE (effectively removing the time component). To convert a DATE-typed UNTIL from iCalendar, set the time components of the LocalDateTime value to "23:59:59". If the iCalendar UNTIL value is a UTC date time, convert it to the local time in the JSCalendar calendar object time zone. To convert to iCalendar where the DTSTART or DUE property is of type DATE, omit the time component of the LocalDateTime value.
relatedTo	RELATED-TO property.
replyTo	An iCalendar ORGANIZER with a mailto: URI mapped to the "imip" method, or any other URI mapped to the "other" method. Mapping multiple methods is

	implementation-specific.
sequence	SEQUENCE property.
showWithoutTime	Implementation-specific.
start	Maps to the DTSTART property. See Section 6.1.
status	STATUS property.
timeZone	Maps to the TZID parameter. See Section 6.1.
timeZones	Each entry in the property maps to a VTIMEZONE in the embedding VCALENDAR component.
title	SUMMARY property.
uid	UID property.
updated	DTSTAMP and LAST-MODIFIED properties. Fractional seconds SHOULD be preserved with the SUBSECOND parameter.
useDefaultAlerts	Implementation-specific.
virtualLocations	See Section 6.2.

Table 4: Translation between JSCalendar and iCalendar

6.1. Time properties and types

iCalendar defines two different time types, DATE and DATE-TIME, where the latter may occur in three forms (with local time, with UTC time, with local time and time zone reference). In contrast, JSCalendar does not define a distinct type for dates, and date times are defined with the LocalDateTime type only.

A JSCalendar time maps to the iCalendar DATE type if all of the following criteria apply:

- o The "start" ("due") property value has zero time, or is not set.
- o The "duration" ("estimatedDuration") property value has zero time, or is a multiple of days or weeks, or is not set.

- o The "timeZone" property value is null, or is not set.

For all other cases, the time maps to an iCalendar DATE-TIME:

- o With local time and time zone reference, if the "timeZone" property value is set and does not equal "Etc/UTC".
- o With UTC time, if the "timeZone" property value equals "Etc/UTC".
- o With local time, if the "timeZone" property value is null or not set.

Fractional seconds SHOULD be preserved with the SUBSECOND parameter.

6.2. Locations

The iCalendar counterpart for JSCalendar Location objects is the iCalendar [RFC5545] LOCATION property, or implementation-specific.

Property	iCalendar counterpart
coordinates	GEO property.
description	Implementation-specific.
linkIds	Implementation-specific.
name	LOCATION property value.
rel	Implementation-specific.
timeZone	Implementation-specific.
uri	The LOCATION ALTREP parameter.

Table 5: Mapping Location properties

The iCalendar counterpart for JSCalendar VirtualLocation objects is the iCalendar [RFC7986] CONFERENCE property.

Property	iCalendar counterpart
description	Implementation-specific.
name	LABEL parameter.
uri	CONFERENCE property value.

Table 6: Mapping virtualLocation properties

6.3. Participants

The following table outlines translation of JSCalendar participants. An iCalendar ORGANIZER maps to both the replyTo property and a participant with role "owner". If an ATTENDEE with the same CAL-ADDRESS value exists, then it maps to the same participant as the ORGANIZER participant. Other participants map to ATTENDEES.

Property	iCalendar counterpart
attendance	ROLE parameter values REQ-PARTICIPANT, OPT-PARTICIPANT and NON-PARTICIPANT.
delegatedFrom	DELEGATED-FROM parameter
delegatedTo	DELEGATED-TO parameter
email	EMAIL parameter, if defined. Otherwise the CAL-ADDRESS property value, if it is a mailto: URI.
expectReply	RSVP parameter
kind	CUTYPE parameter
linkIds	Implementation-specific.
locationId	Implementation-specific.
memberOf	MEMBER parameter
name	CN parameter
participationStatus	PARTSTAT parameter
roles	ROLE parameter.
scheduleSequence	SEQUENCE property of the participant's latest iMIP message
scheduleUpdated	DTSTAMP property of the participant's latest iMIP message
sendTo	A CAL-ADDRESS with a mailto: URI maps to the JSCalendar "imip" method, any other URI to the "other" method. Mapping multiple methods is implementation-specific.

Table 7: Mapping Participant properties

7. Custom properties

Mapping custom or unknown properties between JSCalendar and iCalendar is implementation-specific. Implementations might use vendor-extension properties, which could also serve as basis for discussion for a JSCalendar standard extension. Alternatively, an implementation could preserve iCalendar properties and components in JSCalendar by use of a vendor-extension property formatted as jCal [RFC7265] data.

8. Security Considerations

The same security considerations as for [draft-ietf-calext-jscalendar] apply.

9. IANA Considerations

None.

10. Acknowledgments

The authors would like to thank the members of CalConnect for their valuable contributions. This specification originated from the work of the API technical committee of CalConnect, the Calendaring and Scheduling Consortium.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", RFC 7265, DOI 10.17487/RFC7265, May 2014, <<https://www.rfc-editor.org/info/rfc7265>>.
- [RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <<https://www.rfc-editor.org/info/rfc7986>>.

11.2. Informative References

[draft-apthorp-ical-tasks]
"Task Extensions to iCalendar",
<<https://tools.ietf.org/html/draft-apthorp-ical-tasks>>.

[draft-ietf-calext-ical-relations]
"Support for iCalendar Relationships",
<<https://tools.ietf.org/html/draft-ietf-calext-ical-relations>>.

[draft-ietf-calext-jscalendar]
"Task Extensions to iCalendar",
<<https://tools.ietf.org/html/draft-ietf-calext-jscalendar>>.

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>

Network Working Group
Internet-Draft
Updates: 5988,7240 (if approved)
Intended status: Standards Track
Expires: December 9, 2019

M. Douglass
Spherical Cow Group
June 7, 2019

Calendar subscription upgrades
draft-ietf-calext-subscription-upgrade-00

Abstract

This specification introduces an approach to allow subscribers to calendar feeds to upgrade to a more performant protocol.

This specification updates [RFC5545] to add the value DELETED to the STATUS property.

This specification also updates [RFC7240] to add the subscribe-enhanced-get and limit preferences.

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 9, 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	2
1.1. Terms and Definitions	3
2. Discovering alternative access methods	3
3. Enhanced GET	4
3.1. General	4
3.2. Deletions	5
3.3. Handling of invalid sync tokens	5
3.4. Paging the response	5
3.5. Caching of responses	6
3.6. Examples	6
4. Changes to the iCalendar specifications	8
4.1. Redefined Status property	8
5. Header Field: Sync-Token	10
6. New Prefer header field preferences	10
6.1. Preference subscribe-enhanced-get	10
6.2. Preference limit	10
7. Link relations	10
7.1. General	10
7.2. subscribe-caldav	11
7.3. subscribe-caldav-auth	11
7.4. subscribe-webdav-sync	11
7.5. subscribe-enhanced-get	11
8. Security Considerations	11
9. IANA Considerations	12
9.1. Sync-Token HTTP Header Field Registration	12
9.2. Preference Registrations	12
9.3. Link Relation Registrations	13
10. Acknowledgements	13
11. Normative References	13
Appendix A. Open issues	14
Appendix B. Change log	15
Author's Address	15

1. Introduction

Currently clients subscribe to calendar feeds as an iCalendar file which is often published as a resource accessible using the unofficial 'webcal' scheme.

The only available option for updating that resource is the usual HTTP polling of cached resources using Etags.

There is the usual tension between clients wishing to see a timely response to changes and servers not wishing to be overloaded by frequent requests for possibly large amounts of data.

This specification introduces an approach whereby clients can discover a more performant access method. Given the location of the resource as an iCalendar file, the client can perform a HEAD request on the resource and inspect the returned headers which will offer a number of alternative access methods.

Given that many clients and servers already support CalDAV this provides an easy upgrade path for those clients. CalDAV and DAV subsets are specified here to allow lighter weight implementations.

1.1. Terms and Definitions

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.

Additionally, the rule for URI is included from [RFC3986].

2. Discovering alternative access methods

The advertising of other access points is achieved through the use of the LINK header as defined in [RFC5988]. New link relation types are defined in this specification - each being associated with a protocol or protocol subset.

These LINK headers will be delivered when a client carries out a HEAD request targeting the URL of the resource.

This is an example of a HEAD request and the response from a server that supports the enhanced GET method.

>> Request <<

```
HEAD /caldata/events.ics HTTP/1.1
Host: example.com
Accept: text/calendar
```

>> Response <<

```
HTTP/1.1 200 OK
Content-Length: xxxx
Link: <http://example.com/subscribe/events.ics>;
      rel="subscribe-enhanced-get"
```

Note that the target for an upgraded service may be the same as for the initial resource.

3. Enhanced GET

3.1. General

This is a lightweight protocol which allows simple clients to efficiently discover and download changes in the targeted resource.

It has many similarities to WebDAV sync and for a server could be implemented as an extension of the specification.

In this protocol the client MUST include the Prefer header field preference "subscribe-enhanced-get". If a sync token is available it is passed as a Sync-Token header field.

The resource is treated as a set of individual events each of which may be updated or deleted separately. The client will first fetch the entire iCalendar file. On subsequent requests it uses the Prefer header field and a Sync-Token header field to indicate that it wants a set of changes since the last fetch.

If no Sync-Token header field is supplied the server SHOULD respond with a full set of data. Otherwise, if the token is valid, it SHOULD return with a set of changed entities.

In both cases the server should set the Preference-Applied header field and a new Sync-Token header field value.

3.2. Deletions

When an entity (VEVENT, VTODO or other valid top-level component) is deleted from the source data the server needs to be able to inform a client of the deletion. This specification introduces a new value for the STATUS property of DELETED.

On the first enhanced GET after the entity has been deleted a skeleton, but valid, entity will be returned with STATUS: DELETED. The receiving client is free to remove the entity or update it's STATUS property.

On subsequent fetches the entity will not be returned.

3.3. Handling of invalid sync tokens

When a server receives an invalid token it MUST return a 409 status (Conflict). The server MAY choose to return an error message in the body.

The client SHOULD respond to this error by restarting the interaction from scratch, i.e. retrieve the full set of data then poll for updates.

3.4. Paging the response

A client may explicitly request a limit on the size of the response by specifying the Prefer header field preference "limit=n" where n is the number of components.

When a server receives a request specifying such a limit it SHOULD limit the response to that number of components. If the limit causes a truncation in the response the server should respond with a Preference-Applied header specifying the limit that was applied and return a sync token which may be used to retrieve the next batch of data.

This allows the client to immediately resubmit a request for the next batch using the updated token.

A server MAY choose to limit the response size. The behavior SHOULD be as if the client had provided a preference for that size - allowing the client to retrieve the full set of data in batches.

3.5. Caching of responses

To enable proper caching of responses the server SHOULD provide a VARY header field in responses that names the Prefer and Sync-Token header fields along with any other that are appropriate.

Clients should order the preferences as following so that identical responses can be identified:

- o subscribe-enhanced-get
- o limit

3.6. Examples

This is an example of the initial request and response from a server that supports the enhanced GET method. Note the use of the Vary header so a caching proxy can key off the client's Sync-Token and preference.

>> Request <<

```
GET /events.ics HTTP/1.1
Host: example.com
Accept: text/calendar
Prefer: subscribe-enhanced-get
```

>> Response <<

```
HTTP/1.1 200 OK
Content-Length: xxxx
Sync-Token: "data:,1234567"
Preference-Applied: subscribe-enhanced-get
Vary: Prefer, Sync-Token
```

```
BEGIN:VCALENDAR:
? /* full feed */
END:VCALENDAR
```

This is an example of the subsequent request and response when no changes have occurred.

>> Request <<

```
GET /events.ics HTTP/1.1
Host: example.com
Accept: text/calendar
Prefer: subscribe-enhanced-get
Sync-Token: "data:,1234567"
```

>> Response <<

```
HTTP/1.1 304 Not Modified
Content-Length: 0
Sync-Token: "data:,1234567"
Preference-Applied: subscribe-enhanced-get
Vary: Prefer, Sync-Token
```

This is an example of the subsequent request and response for an old or invalid token.

>> Request <<

```
GET /events.ics HTTP/1.1
Host: example.com
Accept: text/calendar
Sync-Token: "data:,1234567"
Prefer: subscribe-enhanced-get
```

>> Response <<

```
HTTP/1.1 409 Conflict
Content-Length: xxxx
Preference-Applied: subscribe-enhanced-get
```

This is an example of the subsequent request and response when changes have occurred.

>> Request <<

```
GET /events.ics HTTP/1.1
Host: example.com
Accept: text/calendar
Sync-Token: "data:,1234567"
Prefer: subscribe-enhanced-get
```

>> Response <<

```
HTTP/1.1 200 OK
Content-Type: text/calendar
Vary: Prefer, Sync-Token
Sync-Token: "data:,4567890"
Preference-Applied: subscribe-enhanced-get

BEGIN:VCALENDAR:
... only new/changed events
... deleted events have STATUS:DELETED
END:VCALENDAR
```

4. Changes to the iCalendar specifications

This specification updates [RFC5545] to add the value DELETED to the STATUS property.

4.1. Redefined Status property

Property name
STATUS

Purpose
This property defines the overall status or confirmation for the calendar component.

Value Type
TEXT

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

Conformance
This property can be specified once in "VEVENT", "VTODO", or "VJOURNAL" calendar components.

Description

In a group-scheduled calendar component, the property is used by the "Organizer" to provide a confirmation of the event to the "Attendees". For example in a "VEVENT" calendar component, the "Organizer" can indicate that a meeting is tentative, confirmed, or cancelled. In a "VTODO" calendar component, the "Organizer" can indicate that an action item needs action, is completed, is in process or being worked on, or has been cancelled. In a "VJOURNAL" calendar component, the "Organizer" can indicate that a journal entry is draft, final, or has been cancelled or removed.

Format Definition

This property is defined by the following notation:

```
status          = "STATUS" statparam ":" statvalue CRLF

statparam       = *(";" other-param)

statvalue       = (statvalue-event
                  /  statvalue-todo
                  /  statvalue-jour)

statvalue-event = "TENTATIVE"      ;Indicates event is tentative.
                  / "CONFIRMED"    ;Indicates event is definite.
                  / "CANCELLED"    ;Indicates event was cancelled.
                  / "DELETED"      ;Indicates event was deleted.
;Status values for a "VEVENT"

statvalue-todo  = "NEEDS-ACTION" ;Indicates to-do needs action.
                  / "COMPLETED"  ;Indicates to-do completed.
                  / "IN-PROCESS"  ;Indicates to-do in process of.
                  / "CANCELLED"    ;Indicates to-do was cancelled.
                  / "DELETED"      ;Indicates to-do was deleted.
;Status values for "VTODO".

statvalue-jour  = "DRAFT"          ;Indicates journal is draft.
                  / "FINAL"        ;Indicates journal is final.
                  / "CANCELLED"    ;Indicates journal is removed.
                  / "DELETED"      ;Indicates journal was deleted.
;Status values for "VJOURNAL".
```

Example

The following is an example of this property for a "VEVENT" calendar component:

```
STATUS:TENTATIVE
```

The following is an example of this property for a "VTODO" calendar component:

STATUS:NEEDS-ACTION

The following is an example of this property for a "VJOURNAL" calendar component:

STATUS:DRAFT

5. Header Field: Sync-Token

This specification defines a new header field Sync-Token for use by the enhanced GET method.

Accept = DQUOTE URI DQUOTE

The value MUST be a URI. This will generally be a data URI representing an opaque token. Client MUST not attempt to interpret the data URI value.

This is an example of the Sync-Token header field:

Sync-Token: "data:,1234567"

6. New Prefer header field preferences

6.1. Preference subscribe-enhanced-get

This indicates that the client expects the server to handle the GET method according to the specifications for enhanced get.

pref-subscribe-enhanced-get = "subscribe-enhanced-get"

6.2. Preference limit

This preference parameter provides a limit on the number of components returned for enhanced get.

pref-limit = "limit" BWS "=" BWS 1*DIGIT

7. Link relations

7.1. General

This clause defines a number of new link relations required to facilitate subscription upgrades.

7.2. subscribe-caldav

This specifies an access point which is a full implementation of caldav but requires no authentication. The end point allows the full range of reports as defined by the CalDAV specification.

The client **MUST** follow the specification to determine exactly what operations are allowed on the access point - for example to determine if sync-report is supported.

The URL **MAY** include some form of token to allow write access to the targeted collection. The client must check it's permissions to determine whether or not it has been granted write access.

7.3. subscribe-caldav-auth

This specifies an access point which is a full implementation of caldav and requires authentication. This may allow read-write access to the resource.

The client **MUST** follow the specification to determine exactly what operations are allowed on the access point -- for example to determine if sync-report is supported.

7.4. subscribe-webdav-sync

This specifies an access point which supports only webdav sync.

This allows the client to issue a sync-report on the resource to obtain updates.

The client **MUST** follow that specification.

7.5. subscribe-enhanced-get

This specifies an access point which supports something new.

The client **MUST** follow that specification.

8. 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. == 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].

9. IANA Considerations

9.1. Sync-Token HTTP Header Field Registration

This specification updates the "Message Headers" registry entry for "Sync-Token" in [RFC3864] to refer to this document.

Header Field Name: Sync-Token
Protocol: http
Status: standard
Reference: <this-document>

9.2. Preference Registrations

The following preferences have been added to the HTTP Preferences Registry defined in [RFC7240]

Preference
 subscribe-enhanced-get

Value
 None.

Description
 Marks the interaction as enhanced get and provides the optional sync-token and page size.

Reference
 this document

Preference
 limit

Value
 An integer page size.

Description
 Provide a limit on the number of components in the response.

Reference
 this document

9.3. Link Relation Registrations

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

Relation Name	Description	Reference
subscribe-caldav	Current	Section 7.2
subscribe-caldav_auth	Current	Section 7.3
subscribe-webdav-sync	Current	Section 7.4
subscribe-enhanced_get	Current	Section 7.5

10. Acknowledgements

The author would also like to thank the members of the CalConnect Calendar Sharing technical committee and the following individuals for contributing their ideas and support:

Marten Gajda, Ken Murchison, Garry Shutler

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

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, DOI 10.17487/RFC2434, October 1998, <<https://www.rfc-editor.org/info/rfc2434>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <<https://www.rfc-editor.org/info/rfc3864>>.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4589] Schulzrinne, H. and H. Tschofenig, "Location Types Registry", RFC 4589, DOI 10.17487/RFC4589, July 2006, <<https://www.rfc-editor.org/info/rfc4589>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.
- [RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <<https://www.rfc-editor.org/info/rfc5988>>.
- [RFC6578] Daboo, C. and A. Quillaud, "Collection Synchronization for Web Distributed Authoring and Versioning (WebDAV)", RFC 6578, DOI 10.17487/RFC6578, March 2012, <<https://www.rfc-editor.org/info/rfc6578>>.
- [RFC7240] Snell, J., "Prefer Header for HTTP", RFC 7240, DOI 10.17487/RFC7240, June 2014, <<https://www.rfc-editor.org/info/rfc7240>>.
- [RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <<https://www.rfc-editor.org/info/rfc7991>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Appendix A. Open issues

Vary

Ensure we get that right.

Appendix B. Change log

calext00 2019-06-05 MD

- o First calext version
- o Use Sync-Token header rather than parameter

v04 2019-03-07 MD

- o Reference to RFC 6538 - WebDAV sync and RFC 7240 Prefer
- o Go back to HEAD
- o New Preference and parameters.
- o Examples
- o More text for extended get. Talk about deletions.

v01 2017-02-17 MD

- o Add text about OPTIONS
- o Add text about read/write CalDAV

v00 2017-02-15 MD

- o First pass

Author's Address

Michael Douglass
Spherical Cow Group
226 3rd Street
Troy 12180
United States of America

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

Network Working Group
Internet-Draft
Updates: 5545 (if approved)
Intended status: Standards Track
Expires: December 12, 2019

C. Daboo
Apple
K. Murchison, Ed.
FastMail
June 10, 2019

VALARM Extensions for iCalendar
draft-ietf-calext-valarm-extensions-00

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 December 12, 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	2
2.	Conventions Used in This Document	3
3.	Extensible syntax for VALARM	3
4.	Alarm Unique Identifier	5
5.	Alarm Acknowledgement	5
5.1.	Acknowledged Property	6
6.	Snoozing Alarms	7
7.	Alarm Proximity Trigger	8
7.1.	Proximity Property	9
7.2.	Example	10
8.	Security Considerations	10
9.	IANA Considerations	10
9.1.	Property Registrations	10
9.2.	Proximity Value Registry	11
10.	Acknowledgments	11
11.	References	11
11.1.	Normative References	11
11.2.	Informative References	12
11.3.	URIs	12
Appendix A. Change History (To be removed by RFC Editor before publication)		12
Authors' Addresses		14

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,
    ; but MUST NOT occur more than once

    action / trigger /

    ; one set of action properties MUST be
    ; present and MUST match the action specified
    ; in the ACTION property

    actionprops /
```

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

        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 be 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-calex-ext-eventpub-ext] - 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

```
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
```


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]:

Property	Status	Reference
ACKNOWLEDGED	Current	RFCXXXX, Section 5.1
PROXIMITY	Current	RFCXXXX, Section 7.1

9.2. Proximity Value Registry

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

Value	Status	Reference
ARRIVE	Current	RFCXXXX, Section 7.1
DEPART	Current	RFCXXXX, Section 7.1
CONNECT	Current	RFCXXXX, Section 7.1
DISCONNECT	Current	RFCXXXX, Section 7.1

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

- [I-D.ietf-calext-eventpub-extensions]
Douglass, M., "Event Publishing Extensions to iCalendar", draft-ietf-calext-eventpub-extensions-13 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <<https://www.rfc-editor.org/info/rfc4791>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.

- [RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <<https://www.rfc-editor.org/info/rfc5870>>.
- [RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <<https://www.rfc-editor.org/info/rfc7986>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

11.2. Informative References

- [BTcore] Bluetooth Special Interest Group, "Bluetooth Core Specification Version 5.0", December 2016, <<https://www.bluetooth.com/specifications/bluetooth-core-specification>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.

11.3. URIs

- [1] <https://tools.ietf.org/html/bcp14>

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

Changes in ietf-00:

1. Submitted as CALEXT draft.

Changes in daboo-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.
12. Editorial fixes.

Changes in daboo-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 daboo-03: none - resubmission of -02

Changes in daboo-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 daboo-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/>