Network Working Group
Internet-Draft
Updates: 5545 (if approved)
Intended status: Standards Track
Expires: July 4, 2021

M. Douglass
Bedework

December 31, 2020

Support for iCalendar Relationships
draft-ietf-calext-ical-relations-06

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 4, 2021.

Copyright Notice

Copyright (c) 2020 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

iCalendar entities often need to be related to each other or to associated meta-data. The specifications below support relationships of the following forms:

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

Grouped iCalendar: iCalendar entities can be related to each other as a group. CATEGORIES are often used for this purpose but are problematic for application developers due to their lack of consistency and use as a free-form tag.

Linked: Entities can be linked to other entities such as vcards through a URI and associated REL and FMTTYPE parameters.

1.1. Structured iCalendar relationships

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

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

1.2. Grouped iCalendar relationships

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

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

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

1.3. Concept relationships

The name CONCEPT is used by the Simple Knowledge Organization System defined in [W3C.CR-skos-reference-20090317]. The term "concept" more accurately defines what we often mean by a category. It’s not the text string that is important but the meaning attached to it. For example, the term "football" can mean very different sports.

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

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

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

1.4. Linked relationships

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

This document proposes a method for referencing typed external information that can provide additional information about an iCalendar component. This new LINK property is closely aligned to the LINK header defined in [RFC8288]

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

It is also often necessary to reference calendar components in other collections. For example, a VEVENT might refer to a VTODO from which it was derived. The PARENT, SIBLING and CHILD relationships defined for the RELATED-TO property only allow for a UID which is inadequate for many purposes. Allowing other value types for those relationships may help but would cause backward compatibility issues. The link property can link components in different collections or even on different servers.

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

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

The LINK property SHOULD NOT be treated as just another attachment. The ATTACH property is being extended to handle server-side management and stripping of inline data. Clients may choose to handle attachments differently as they are often an integral part of

the message - for example, the agenda. See [I-D.daboo-caldav-attachments]

1.5. Caching and offline use

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

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

1.6. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Reference Types

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

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

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

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

3. Link Relation Types

[RFC8288] defines two forms of relation type: registered and extension. Registered relation types are added to the Link Relations registry as specified in Section 2.1.1 of [RFC8288]. Extension relation types, defined in Section 2.1.2 of [RFC8288], are specified as unique URIs that are not registered in the registry.

The relation types defined in Section 6.1 will be registered with IANA in accordance with the specifications in [RFC8288].

4. New temporal RELTYPE Parameter values

This section defines the usual temporal relationships for use with the RELTYPE parameter redefined in Section 3.2.15 of [RFC5545]: FINISHTOSTART, FINISHTOFINISH, STARTTOFINISH or STARTTOSTART.

The [RFC5545] RELATED-TO property with one or more of these temporal relationships will be present in the predecessor entity and will refer to the successor entity.

The GAP parameter (see Section 6.2) specifies the lead or lag time between the predecessor and the successor. In the description of each temporal relationship below we refer to Task-A, which contains and controls the relationship, and Task-B the target of the relationship.

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

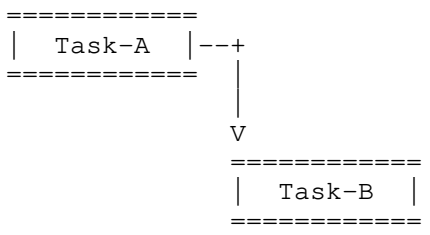

Figure 1: Finish to start relationship

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

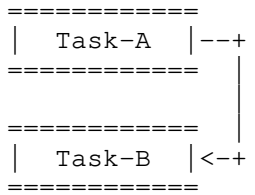

Figure 2: Finish to finish relationship

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

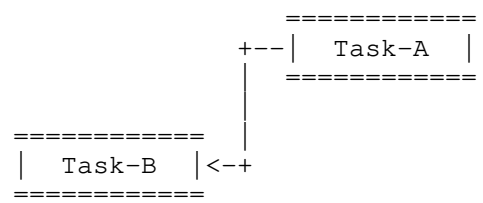

Figure 3: Start to finish relationship

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

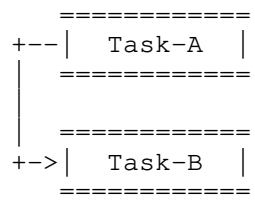

Figure 4: Start to start relationship

5. Additional New RELTYPE Parameter Values

This section defines the additional relationships below:

RELTYPE=DEPENDS-ON: Indicates that the current calendar component depends on the referenced calendar component in some manner. For example a task may be blocked waiting on the other, referenced, task.

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

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

6. New Property Parameters

6.1. Link Relation

Parameter name: LINKREL

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

Format Definition:

This parameter is defined by the following notation:

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

Description: This parameter MUST be specified on all LINK properties, and defines the type of reference. This allows programs consuming this data to automatically scan for references they support. In addition to the values defined here any value defined in [RFC8288] may be used. There is no default relation type.

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

Registration: These relation types are registered in [RFC8288]

6.2. Gap

Parameter name: GAP

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

Format Definition:

This parameter is defined by the following notation:

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

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

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

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

7. New Value Data Types

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

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

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

8. New Properties

8.1. Concept

Property name: CONCEPT

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

Value type: URI

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

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

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

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

This categorization is distinct from the more informal "tagging" of components provided by the existing CATEGORIES property. It is

expected that the value of the CONCEPT property will reference an external resource which provides information about the categorization.

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

Possible category resources are the various proprietary systems, for example Library of Congress, or an open source of categorisation data.

Format Definition:

This property is defined by the following notation:

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

conceptparam = *(";" other-param)
```

Example:

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

```
CONCEPT:http://example.com/event-types/arts/music
```

8.2. Link

Property name: LINK

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

Value type: URI, TEXT or REFERENCE

Property Parameters: The VALUE parameter is required. Non-standard, reference type or format type parameters can also be specified on this property. The LABEL parameter is defined in [RFC7986]

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

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

Format Definition:

This property is defined by the following notation:

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

linkparam      = ; the elements herein may appear in any order,
                 ; and the order is not significant.

                 (";" "VALUE" "=" ("UID" /
                                   "URI" /
                                   "TEXT"))
                 1*(";" linkrelparam)
                 (";" fmttypeparam)
                 (";" labelparam)
                 *(";" other-param)
```

Example:

The following is an example of this property which provides a reference to the source for the calendar object.

```
LINK;LINKREL=SOURCE;LABEL=Venue;VALUE=URI:http://example.com/events
```

Example:

The following is an example of this property which provides a reference to an entity from which this one was derived. The link relation is a vendor defined value

```
LINK;LINKREL="https://example.com/linkrel/derivedFrom";VALUE=URI:
 http://example.com/tasks/01234567-abcd1234.ics
```

8.3. Refid

Property name: REFID

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

Value type: TEXT

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

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

Description: The value of this property is free-form text that creates an identifier for associated components. All components that use the same REFID value are associated through that value and can be located or retrieved as a group. For example, all of the events in a travel itinerary would have the same REFID value, so as to be grouped together.

Format Definition:

This property is defined by the following notation:

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


refidparam      = *(";" other-param)
```

Example:

The following is an example of this property.

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

9. Redefined RELATED-TO Property

9.1. RELATED-TO

Property name: RELATED-TO

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

Value type: URI, UID or TEXT

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

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

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

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

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

To preserve backwards compatibility the value type MUST be UID when the PARENT, SIBLING or CHILD relationships are specified.

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

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

Format Definition:

This property is defined by the following notation:

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

relparam   = ; the elements herein may appear in any order,
             ; and the order is not significant.
             [";" "VALUE" "=" ("UID" /
                               "URI" /
                               "TEXT")]
             [";" reltypeparam]
             [";" gapparam]
             *(";" other-param)
```

Example:

The following are examples of this property.

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

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

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

10. Security Considerations

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

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

11. IANA Considerations

11.1. iCalendar Property Registrations

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

Property	Status	Reference
CONCEPT	Current	Section 8.1
LINK	Current	Section 8.2
REFID	Current	Section 8.3

11.2. iCalendar Property Parameter Registrations

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

Parameter	Status	Reference
REL	Current	Section 6.1
GAP	Current	Section 6.2

11.3. iCalendar Value Data Type Registrations

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

Value Data Type	Status	Reference
UID	Current	Section 7
REFERENCE	Current	Section 7

11.4. iCalendar RELTYPE Value Registrations

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

Relationship Type	Status	Reference
DEPENDS-ON	Current	Section 5
REFID	Current	Section 5
CONCEPT	Current	Section 5
FINISHTOSTART	Current	Section 4
FINISHTOFINISH	Current	Section 4
STARTTOFINISH	Current	Section 4
STARTTOSTART	Current	Section 4

11.5. New Reference Type Registration

The following link relation values have been added to the Reference Types Registry defined in Section 6.2.2 of [RFC8288]

Name	Status	Reference
SOURCE	Current	Section 6.1

12. Acknowledgements

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

Adrian Apthorp, Cyrus Daboo, Marten Gajda, Ken Murchison

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

13. Normative References

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

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

[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>.

[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>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

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

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

Author's Address

Michael Douglass
Bedework
226 3rd Street
Troy, NY 12180
USA

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 April 2021

M. Douglass

14 October 2020

Support for Series in iCalendar
draft-ietf-calext-icalendar-series-02

Abstract

This document updates [RFC5545] by defining a new repeating set of events known as a series. This differs from recurrences in that each instance is a separate entity with a parent relationship to a specified template entity.

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 17 April 2021.

Copyright Notice

Copyright (c) 2020 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. Acknowledgements

The authors would like to thank the members of the Calendaring and Scheduling Consortium (CalConnect) for contributing their ideas and support.

2. Introduction

Since iCalendar was first defined there has been only one way to express a repeating set of events - the recurrence. This defined a master event, a set of rules for computing the instances and a way of overriding certain instances.

This approach works well enough in certain situations but has many problems which need to be addressed.

This specification introduces a new approach to repeating patterns of entities which avoids some of the problems.

3. Terms and definitions

No terms and definitions are listed in this document.

4. Overrides and iCalendar recurrences

The recurrence rules specify how instances are to be computed. These rules provide a set of keys - the RECURRENCE-ID - and an instance can be created with the calculated start date/time and a copy of the duration (or calculated end date/time).

The specification allows for overrides. These are handled by supplying a complete replacement for the instance with a RECURRENCE-ID property matching that of the instance being overridden. This may change any of the properties (except the UID) - including start, end or duration.

If a long lived recurrence is heavily overridden it becomes very cumbersome. The master plus overrides is considered a single resource in most circumstances (iTip allows the delivery of a single instance in certain situations).

Simple meetings can become heavily modified recurrences through adding the weeks agenda to the description, changing of attendees etc.

There are approaches being considered to mitigate some of these issues which mostly involve only storing changes but recurrences are still awkward to deal with.

4.1. Changing the master start or the recurrence rules

This can lead to some very difficult problems to resolve. In the case of a heavily modified meeting it may be difficult to impossible to determine which override applies to the newly modified event.

For example, a weekly book-reading is moved from Monday to Friday. There are weeks of scheduled events in the future. Do we move them all forward to the next instance or skip one and move them back? If it becomes bi-weekly rather than weekly do we drop every other or just space them out more?

To be sure - some of these problems are not totally resolved by a series approach but they become more tractable.

4.2. Splitting recurrences

The [RFC5545] THISANDFUTURE range is poorly supported. Splitting is what a number of implementations use to avoid changing overrides in the past.

The recurring event is split into 2, one being the truncated original the other being a new recurring event starting at the time of the THISANDFUTURE override.

There is left the problem of relating the two, this can be accomplished by use of the RELATED-TO property but that is not standardized.

5. Series

A series is a, generally regularly, repeating sets of events or tasks each instance of which is usually, but not always, different in some respect. Examples may be a library running an after-school reading program which usually, takes place at the same time each week but always differs in the book or author being studied.

In recurrences an instances is a calculated 'virtual' object, unless overridden. It has the same UID as the master and a RECURRENCE-ID which is always one of the calculated set.

In a series, a specified number of instances are created ahead of time each with their own unique UID. They are all related to the master using a SERIES-MASTER relation type defined in this specification. Each instance acts as an individual component as far as retrieval and searching is concerned.

Each instance and master is identified as a member of the full series by the SERIES-UID property. The value of this property is the same in all members of the series even when splits have occurred.

As instances are created a LAST-SERIES-ID property is added or updated in the master to indicate which instance was last created. When there are SXDATE properties this property value may represent an instance which cannot be created. It merely represents the latest calculated date.

This property allows generated instances to be deleted without the addition of SXDATE properties to the master. The SXDATE only indicates future instances which MUST NOT be created.

As time goes on more instances are created either by the server or by a client when it inspects the current state of the series. The number of instances may be based on time or a count.

For example, an organization may allow rooms to be booked only 4 weeks ahead. Thus a series may be set up which has that 4 week set of events in the future. Each will have the room as an attendee ensuring that at least the room is booked at the regular time.

5.1. Modifying series patterns and splitting

If it becomes necessary to modify the series rules or the master start then the series is always split at the point of the modification.

When a series is split the previous master is modifed to truncate the current series at the last generated instance and a parameter SPLIT=YES is added to the series rule to indicate that this master is now split.

The split may result in a number of instances related to the old series but overlapping the new. It is up to the implementation to decide what should be done with these but this usually requires a degree of interaction with a human (or very intelligent robot). The application may offer to copy them into the corresponding new instances - if these can be easily determined, offer to delete all of them or let the user manually copy information and delete.

The new series master is related to the old master by the new series master having a RELATED-TO property with RELTYPE=SERIES-MASTER pointing at the previous master. In that way a backwards chain of series masters may be created

5.2. The series master

A series master is identified in much the same way as a recurrence master. It will contain an SRULE and 0 or more SDATE properties or 1 or more SDATE properties. Additionally it may contain 0 or more SXDATE properties to exclude instances.

As noted above, if the series was split it may contain a RELATED-TO property with RELTYPE=SERIES-MASTER and a value of the previous series master.

The master will also contain a LAST-SERIES-ID if any instances have been calculated and perhaps generated.

It is important to note that the series master is the first member of the series. Thus the first instance always occurs AFTER the series master.

5.3. The series instances

A series instance is identified by having a SERIES-ID property which is calculated in the same manner as a RECURRENCE-ID. It MUST also contain a RELATED-TO property with RELTYPE=SERIES-MASTER and a value being the UID of the series master.

As noted above, if the series was split it may contain a RELATED-TO property with RELTYPE=SERIES-MASTER and a value being the UID of the previous series master.

6. Redefined Relation Type Value

Relationship parameter type values are defined in [RFC5545]. This specification augments that parameter to include the new relationship values SERIES-MASTER.

Format Definition

This property parameter is respecified as follows:

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

Figure 1

Description

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

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

RELTYPE=PARENT See [RFC5545].

RELTYPE=CHILD See [RFC5545].

RELTYPE=SIBLING See [RFC5545].

RELTYPE=DEPENDS-ON Indicates that the current calendar component

depends on the referenced calendar component in some manner. For example a task may be blocked waiting on the other, referenced, task.

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

RELTYPE=SERIES-MASTER Indicates that the current calendar component is based on the referenced calendar component. The value is a UID.

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

RELTYPE=FINISHTOSTART

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

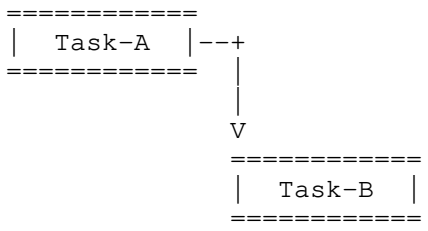

Figure 2: Finish to start relationship

RELTYPE=FINISHTOFINISH

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

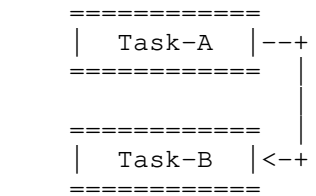

Figure 3: Finish to finish relationship

RELTYPE=STARTTOFINISH

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

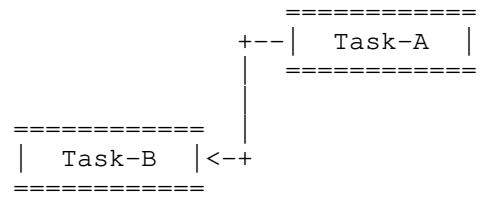

Figure 4: Start to finish relationship

RELTYPE=STARTTOSTART

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

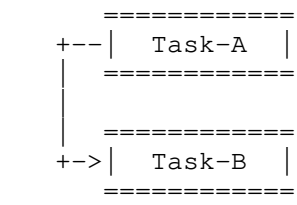

Figure 5: Start to start relationship

7. New Property Parameters

7.1. Split

Parameter name SPLIT

Purpose To indicate a series has been split.

Format Definition

This parameter is defined by the following notation:

```
splitparam     = "SPLIT" "="
                ("YES"      ; The series is split
                / "NO"      ; The series is not split (default)
                / x-name        ; Experimental reference type
                / iana-token)   ; Other IANA registered type
```

Figure 6

Description This parameter MAY be specified on the SRULE property to indicate that the series has been split with SPLIT=YES. Once split is is probably inappropriate to modify the series further.

7.2. Lookahead count

Parameter name LOOKAHEAD-COUNT

Purpose To specify the number of series instances that should be generated in advance.

Format Definition

This parameter is defined by the following notation:

```
lookahead-countparam      = "LOOKAHEAD-COUNT" "=" 1*DIGIT
```

Figure 7

Description

This parameter MAY be specified on the SRULE property to indicate how many series instances should be generated in advance.

An implementation is free to apply its own limts but MUST NOT generate more than those defined by this parameter and/or the LOOKAHEAD-PERIOD parameter.

If both the LOOKAHEAD-PERIOD and LOOKAHEAD-COUNT arameters are supplied the result should be limited by both.

For example, if the LOOKAHEAD-PERIOD parameter would cause 8 instances to be generated but LOOKAHEAD-COUNT specifies 4 then only 4 instances will be generated.

7.3. Lookahead period

Parameter name LOOKAHEAD-PERIOD

Purpose To specify a maximum period for which series instances should be generated in advance.

Format Definition

This parameter is defined by the following notation:

```
lookahead-periodparam      = "LOOKAHEAD-PERIOD" "="
                      DQUOTE dur-value DQUOTE
```

Figure 8

Description

This parameter MAY be specified on the SRULE property to indicate how far in advance series instances should be generated.

An implementation is free to apply its own limts but MUST NOT generate more than those defined by this parameter and/or the LOOKAHEAD-COUNT parameter.

If both the LOOKAHEAD-PERIOD and LOOKAHEAD-COUNT arameters are supplied the result should be limited by both.

For example, if the LOOKAHEAD-PERIOD parameter would cause 8 instances to be generated but LOOKAHEAD-COUNT specifies 4 then only 4 instances will be generated.

The value is a quoted duration.

8. New Properties

8.1. General

The SERIES-ID, LAST-SERIES-ID, SDATE and SXDATE properties are identical in form and in the parameters they take.

All must conform in form to the DTSTART property of the master component. Only the SDATE may specify a time which is not part of the calculated series.

The SRULE property value is identical in form to the RRULE property defined in [RFC5545]. The LOOKAHEAD-COUNT and LOOKAHEAD-PERIOD parameters indicate how many instances should be generated in advance.

8.2. Generating Series members

An agent, either the server or a client, will periodically extend the set of instances. The number of such generated instances is limited by:

Elements of the rule The UNTIL or COUNT parts of the rule define when the series terminates. Thus a COUNT=100 specifies a maximum of 100 series members.

Lookahead count This specifies how many series members can exist from the current date/time. Thus a LOOKAHEAD-COUNT=4 means a maximum of 4 generated instances.

Lookahead period This specifies how far into the future series members can be generated. Thus a LOOKAHEAD-PERIOD="PT2M" means a maximum period of 2 months.

System limits This client or server SHOULD also apply limits to prevent a series from generating an overlarge set of members.

The starting point for the calculation is the DTSTART of the master component or the LAST-SERIES-ID if it exists in the master. In both cases the instance represented by that date is NOT generated as part of the instance set and the actual instance may have been excluded by an SXDATE property but the starting date is still valid.

The starting date/time property defines the first instance in the next batch of series members. Note that the starting property value MUST match the pattern of the series rule, if specified. For example, if the rule specifies every Wednesday the starting date MUST be a Wednesday.

The end date/time of the set will be provided by the UNTIL part of the rule, the LOOKAHEAD-PERIOD or by a system maxima.

A set of date/time values can be generated within those contraints. As each date/time value is generated it can be ignored if it is one of the SXDATE values.

Generation of values can terminate when the size of the result exceeds that given by the COUNT rule element, the LOOKAHEAD-COUNT value or any system limit.

Any SDATE values that fall within the current range and are not in the set of SXDATE values can be added and the result truncated again to match the size limits.

Finally, any date/time values that have already been generated and are present as SERIES-ID values should be removed from the set. What remains is the new set of members to extend the current series.

The last of those values becomes the new value for the LAST-SERIES-ID property in the series master.

As noted above the "SXDATE" property can be used to exclude the value specified in the master. This leads to a complication as the master needs to be preserved as a container for the values which define the series. This is flagged by adding a DELETED-MASTER element to the SERIES-STATUS property.

8.3. Series UID

Property name SERIES-UID

Purpose This property defines the persistent, globally unique identifier for the full series.

Value Type TEXT

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

Conformance This property MUST be specified in any "VEVENT", "VTODO", and "VJOURNAL" calendar components acting as a series master or series instance.

Description The SERIES-UID MUST be globally unique. This value SHOULD be generated by following the recommendations in [RFC7986].

Format Definition

This property is defined by the following notation:

```
seruid        = "SERIES-UID" seruidparam ":" text CRLF

seruidparam   = *(";" other-param)
```

Figure 9

EXAMPLE

The following is an example of this property:

```
SERIES-UID:123e4567-e89b-12d3-a456-426655440000
```

Figure 10

8.4. Series-exception-date

Property name SXDATE

Purpose This property defines the list of DATE-TIME exceptions for

series of events, to-dos or journal entries.

Value Type The default value type for this property is DATE-TIME. The value type can be set to DATE.

Property Parameters IANA, non-standard, value data type, and time zone identifier property parameters can be specified on this property.

Conformance This property can be specified in "VEVENT", "VTODO", and "VJOURNAL" calendar components acting as the series master.

Description The exception dates, if specified, are used when computing the instances of the series. They specify date/time values which are to be removed from the set of possible series instances.

Format Definition

This property is defined by the following notation:

```
sxdate     = "SXDATE" sxdtparam ":" sxdtval *("," sxdtval) CRLF

sxdtparam  = *(
           ;
           ; The following are OPTIONAL,
           ; but MUST NOT occur more than once.
           ;
           (";" "VALUE" "=" ("DATE-TIME" / "DATE")) /
           ;
           (";" tzidparam) /
           ;
           ; The following is OPTIONAL,
           ; and MAY occur more than once.
           ;
           (";" other-param)
           ;
           )

sxdtval    = date-time / date
           ;Value MUST match value type
```

Figure 11

EXAMPLE

The following is an example of this property:

```
SXDATE:19960402T010000Z,19960403T010000Z,19960404T010000Z
```

Figure 12

8.5. Series-date

Property name SDATE

Purpose This property defines the list of DATE-TIME values for series of events, to-dos or journal entries.

Value Type The default value type for this property is DATE-TIME. The value type can be set to DATE.

Property Parameters IANA, non-standard, value data type, and time zone identifier property parameters can be specified on this property.

Conformance This property can be specified in "VEVENT", "VTODO", and "VJOURNAL" calendar components acting as the series master.

Description This property can appear along with the "SRULE" property to define a extra series occurrences. When they both appear in a series master component, the instances are defined by the union of occurrences defined by both the "SDATE" and "SRULE".

Format Definition

This property is defined by the following notation:

```
    sdate      = "SDATE" sdtparam ":" sdtval *("," sdtval) CRLF

    sdtparam   = *(
                ;
                ; The following are OPTIONAL,
                ; but MUST NOT occur more than once.
                ;
                (";" "VALUE" "=" ("DATE-TIME" / "DATE" / "PERIOD")) /
                (";" tzidparam) /
                ;
                ; The following is OPTIONAL,
                ; and MAY occur more than once.
                ;
                (";" other-param)
                ;
                )

    sdtval     = date-time / date
                ;Value MUST match value type
```

Figure 13

EXAMPLE

The following are examples of this property:

```
SDATE:19970714T123000Z
SDATE;TZID=America/New_York:19970714T083000

SDATE;VALUE=PERIOD:19960403T020000Z/19960403T040000Z,
 19960404T010000Z/PT3H

SDATE;VALUE=DATE:19970101,19970120,19970217,19970421
 19970526,19970704,19970901,19971014,19971128,19971129,19971225
```

Figure 14

8.6. Series-id

Property name SERIES-ID

Purpose This property is used in conjunction with the "UID" and "SEQUENCE" properties to identify a specific instance of a "VEVENT", "VTODO", or "VJOURNAL" calendar component in a series. The property value is the original value of the "DTSTART" property of the series instance before any changes occur.

Value type The default value type is DATE-TIME. The value type can

be set to a DATE value type. This property MUST have the same value type as the "DTSTART" property contained within the series component. Furthermore, this property MUST be specified as a date with local time if and only if the "DTSTART" property contained within the series component is specified as a date with local time.

Property Parameters IANA, non-standard, value data type and time zone identifier parameters can be specified on this property.

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

Description

The SERIES-ID is the originally calculated value of the DTSTART property based on the master identified by the RELATED-TO property with a RELTYPE=SERIES-MASTER parameter.

The full series of components can only be retrieved by searching for all components with a matching RELATED-TO property.

Figure 15

If the value of the "DTSTART" property is a DATE type value, then the value MUST be the calendar date for the series instance.

Figure 16

The DATE-TIME value is set to the time when the original series instance would occur; meaning that if the intent is to change a Friday meeting to Thursday, the DATE-TIME is still set to the original Friday meeting.

Figure 17

The "SERIES-ID" property is used in conjunction with the "UID" and "SEQUENCE" properties to identify a particular instance of an event, to-do, or journal in the series. For a given pair of "UID" and "SEQUENCE" property values, the "SERIES-ID" value for a series instance is fixed.

Figure 18

Format Definition

This property is defined by the following notation:

```
   serid    = "SERIES-ID" sidparam ":" sidval CRLF

   sidparam   = *(
                  ;
                  ; The following are OPTIONAL,
                  ; but MUST NOT occur more than once.
                  ;
                  (";" "VALUE" "=" ("DATE-TIME" / "DATE")) /
                  (";" tzidparam) /
                  ;
                  ; The following is OPTIONAL,
                  ; and MAY occur more than once.
                  ;
                  (";" other-param)
                  ;
                  )

   sidval     = date-time / date
                  ;Value MUST match value type
```

Figure 19

EXAMPLE

The following are examples of this property:

```
SERIES-ID;VALUE=DATE:19960401

SERIES-ID;TZID=America/New_York:20170120T120000
```

Figure 20

8.7. Last series ID

Property name LAST-SERIES-ID

Purpose

To specify the last calculated instance of the series. When new instances are created they MUST have a SERIES-ID after the value of this property.

In all respects this property is identical to SERIES-ID and is in fact a copy of the SERIES-ID which would be present in the last created instance (assuming it is not suppressed by an SXDATE).

Value type DATE or DATE_TIME (the default). This has the same requirements as SERIES-ID.

Property Parameters IANA, non-standard, value data type and time zone identifier parameters can be specified on this property.

Conformance This property MAY be specified in any iCalendar component.

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

Format Definition

This property is defined by the following notation:

```
last-series-i   = "LAST-SERIES-ID" lastseriesidparam  /
                  (
                    ";" "VALUE" "=" "TEXT"
                    ":" text
                  )
                  (
                    ";" "VALUE" "=" "REFERENCE"
                    ":" text
                  )
                  (
                    ";" "VALUE" "=" "URI"
                    ":" uri
                  )
                  CRLF


lastseriesidparam = *(

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

                (";" relparam) /

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

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

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

                (";" xparam)

                )
```

Figure 21

EXAMPLE

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

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

Figure 22

8.8. Series Rule

Property name RRULE

Purpose This property defines a rule or repeating pattern for a series of events, to-dos or journal entries.

Value Type RECUR

Property Parameters IANA, non-standard, look-ahead count or date property parameters can be specified on this property.

Conformance This property can be specified in any "VEVENT", "VTODO", and "VJOURNAL" calendar component, but it SHOULD NOT be specified more than once.

Description

The series rule, if specified, is used in computing the instances to be generated for the series. These are generated by considering the master "DTSTART" property along with the "SRULE", "SDATE", and "SXDATE" properties contained within the series master. The "DTSTART" property defines the first instance in the recurrence set which is represented by that master event.

Unlike the RRULE the "DTSTART" property MUST be synchronized with the series rule, if specified. For example, if the DTSTARTS species a date on Wednesday but the SRULE specifies every Tuesday then a server or client MUSt reject the component.

The final series is represented by gathering all of the start DATE-TIME values generated by any of the specified "SRULE" and "SDATE" properties, and then excluding any start DATE-TIME values specified by "SXDATE" properties. This implies that start DATE- TIME values specified by "SXDATE" properties take precedence over those specified by inclusion properties (i.e., "SDATE" and "SRULE"). Where duplicate instances are generated by the "SRULE" and "SDATE" properties, only one instance is considered. Duplicate instances are ignored.

The "DTSTART" property specified within the master iCalendar object defines the first instance of the recurrence. In most cases, a "DTSTART" property of DATE-TIME value type used with a series rule, should be specified as a date with local time and time zone reference to make sure all the recurrence instances start at the same local time regardless of time zone changes.

If the duration of the series component is specified with the "DTEND" or "DUE" property, then the same exact duration will apply to all the members of the generated series. Else, if the duration of the series master component is specified with the "DURATION" property, then the same nominal duration will apply to all the members of the generated series and the exact duration of each instance will depend on its specific start time. For example, series instances of a nominal duration of one day will have an exact duration of more or less than 24 hours on a day where a time zone shift occurs. The duration of a specific instance may be modified in an exception component or simply by using an "SDATE" property of PERIOD value type.

Format Definition

This property is defined by the following notation:

```
srule      = "SRULE" srulparam ":" recur CRLF

sruleparam = *(
             ; the following are OPTIONAL
             ; but MUST NOT occur more than once

             (";" lookahead-countparam) /
             (";" lookahead-periodparam) /

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

             (";" xparam)

             )
```

Figure 23

EXAMPLE

TODO - Say they are pretty much the same as RRULE but extra params

9. Redefined RELATED-TO Property

9.1. RELATED-TO

Property name RELATED-TO

Purpose This property is used to represent a relationship or reference between one calendar component and another. The definition here extends the definition in [RFC5545] by including a section on RELTYPE=SERIES-MASTER.

Value type URI, UID or TEXT

Conformance This property MAY be specified in any iCalendar component.

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

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

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

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

The SERIES-MASTER relationship when included in a series instance refers to the master of that series. When included in a series master it refers to a previous master in a chain of spilt series.

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

Format Definition This property is defined by the following notation:

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

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

Figure 24

EXAMPLE

The following are examples of this property.

```
RELATED-TO;RELTYPE=SERIES-MASTER:19960401-080045-4000F192713
```

Figure 25

10. Backwards compatibility

Any clients following the approach specified in [RFC5545] are expected to ignore any properties or parameters they don’t recognize.

For such clients the series appears to be an unconnected set of components. They all have their own unique UIDS. If the client updates an instance this should be identical in effect to an update carried out by a client aware of the new properties.

Updates MUST preserve the SERIES-ID, LAST-SERIES-ID, SRULE, SDATE and SXDATE properties. A client which does not do so is in violation of [RFC5545].

TODO - More text needed here...

11. CalDAV extensions

This specification may extend Caldav by adding reports to return all members of a series given the series master UID. This could be handled by the current query mechanism but it is likely to be sufficiently frequently used that a special query is appropriate.

It is also likely we will want a CalDAV operation to split a series and generate the additional members of the series as a single atomic operation. == Security Considerations

Clients and servers should take care to limit the number of generated instances to a reasonable value. This can be a relatively small value.

12. IANA Considerations

12.1. iCalendar Property Registrations

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

Property	Status	Reference
LAST-SERIES-ID	Current	Section 8.7
SERIES-ID	Current	Section 8.6
SERIES-UID	Current	Section 8.3
SDATE	Current	Section 8.5
SRULE	Current	Section 8.8
SXDATE	Current	Section 8.4

Table 1

12.2. iCalendar Property Parameter Registrations

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

Parameter	Status	Reference
LOOKAHEAD-COUNT	Current	Section 7.2
LOOKAHEAD-PERIOD	Current	Section 7.3
SPLIT	Current	Section 7.1

Table 2

12.3. iCalendar RELTYPE Value Registrations

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

Relationship Type	Status	Reference
SERIES-ID	Current	Section 5

Table 3

13. Normative references

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

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

Author's Address

Michael Douglass

Email: mikeadouglass@gmail.com

Calendaring extensions
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2021

N. Jenkins
R. Stepanek
Fastmail

October 16, 2020

JSCalendar: A JSON representation of calendar data
draft-ietf-calext-jscalendar-32

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 and, over time, successor to the widely deployed iCalendar data format, and to be unambiguous, extendable, and simple to process. In contrast to the jCal format, which is also JSON-based, JSCalendar is not a direct mapping from iCalendar, but defines the data model independently 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 April 19, 2021.

Copyright Notice

Copyright (c) 2020 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

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

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

- The attributes of the calendar entry represented must be described as simple key-value pairs. Simple events are simple to represent; complex events can be modelled accurately.

- Wherever possible, there should be only one way to express the desired semantics, reducing complexity.

- The data model should avoid ambiguities, which often lead to interoperability issues between implementations.

- The data model should be generally compatible with the iCalendar data format [RFC5545] [RFC7986] and extensions, but the specification should add new attributes where the iCalendar format currently lacks expressivity, and drop seldom-used, obsolete, or redundant properties. This means translation with no loss of semantics should be easy with most common iCalendar files.

- Extensions, such as new properties and components, should not require updates 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, which iCalendar has often suffered from.

1.1. Motivation and Relation to iCalendar and jCal

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.

Sources of implementation errors include the following:

- iCalendar defines various formats for local times, UTC time, and dates.

- iCalendar requires custom time zone definitions within a single calendar component.

- iCalendar's definition of recurrence rules is ambiguous and has resulted in differing understandings even between experienced calendar developers.

- The iCalendar format itself causes interoperability issues due to misuse of CRLF-terminated strings, line continuations, and subtle differences among iCalendar parsers.

In recent years, many new products and services have appeared that wish to use a JSON representation of calendar data within their APIs. The JSON format for iCalendar data, jCal [RFC7265], is a direct mapping between iCalendar and JSON. In its effort to represent full

iCalendar semantics, it inherits all the same pitfalls and uses a complicated JSON structure.

As a consequence, since the standardization of jCal, the majority of implementations and service providers either kept using iCalendar, or came up with their own proprietary JSON representations, which are incompatible with each other and often suffer from common pitfalls, such as storing event start times in UTC (which become incorrect if the timezone's rules change in the future). JSCalendar meets the demand for JSON-formatted calendar data that is free of such known problems and provides a standard representation as an alternative to the proprietary formats.

1.2. Notational Conventions

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.

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.

1.3. Type Signatures

Type signatures are given for all JSON values in this document. The following conventions are used:

- "*" - The type is undefined (the value could be any type, although permitted values may be constrained by the context of this value).
- "String" - The JSON string type.
- "Number" - The JSON number type.
- "Boolean" - The JSON boolean type.
- "A[B]" - A JSON object where the keys are all of type "A", and the values are all of type "B".

- "A[]" - An array of values of type "A".
- "A|B" - The value is either of type "A" or of type "B".

Other types may also be given, with their representations defined elsewhere in this document.

1.4. Data Types

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

1.4.1. Id

Where "Id" is given as a data type, it means a "String" of at least 1 and a maximum of 255 octets in size, and it MUST only contain characters from the "URL and Filename Safe" base64url alphabet, as defined in Section 5 of [RFC4648], excluding the pad character ("="). This means the allowed characters are the ASCII alphanumeric characters ("A-Za-z0-9"), hyphen ("-"), and underscore ("_").

In many places in JSCalendar a JSON map is used where the map keys are of type Id and the map values are all the same type of object. This construction represents an unordered set of objects, with the added advantage that each entry has a name (the corresponding map key). This allows for more concise patching of objects, and, when applicable, for the objects in question to be referenced from other objects within the JSCalendar object.

Unless otherwise specified for a particular property, there are no uniqueness constraints on an Id value (other than, of course, the requirement that you cannot have two values with the same key within a single JSON map). For example, two JSEvent objects might use the same Ids in their respective "links" properties. Or within the same JSEvent object the same Id could appear in the "participants" and "alerts" properties. These situations do not imply any semantic connections among the objects.

Nevertheless, a UUID [RFC4122] is typically a good choice.

1.4.2. Int

Where "Int" is given as a data type, it means an integer in the range -2^53+1 <= value <= 2^53-1, the safe range for integers stored in a floating-point double, represented as a JSON "Number".

1.4.3. UnsignedInt

Where "UnsignedInt" is given as a data type, it means an integer in the range 0 <= value <= 2^53-1, represented as a JSON "Number".

1.4.4. UTCDateTime

This is a string in [RFC3339] "date-time" format, with the further restrictions that any letters MUST be in uppercase, 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 conformant, but "2010-10-10T10:10:10.000Z" is invalid and is correctly encoded as "2010-10-10T10:10:10Z".

1.4.5. 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 with the LocalDateTime comes from the "timeZone" property of the JSCalendar object (see Section 4.7.1). If no time zone is specified, the LocalDateTime is "floating". Floating date-times are not tied to any specific time zone. Instead, they occur in each time zone at the given wall-clock time (as opposed to the same instant point in time).

A time zone may have a period of discontinuity, for example a change from standard time to daylight-savings time. When converting local date-times that fall in the discontinuity to UTC, the offset before the transition MUST be used.

For example, in the America/Los_Angeles time zone, the date-time 2020-11-01T01:30:00 occurs twice: before the DST transition with a UTC offset of -07:00, and again after the transition with an offset of -08:00. When converting to UTC, we therefore use the offset before the transition (-07:00) and so it becomes 2020-11-01T08:30:00Z.

Similarly, in the Australia/Melbourne time zone, the date-time 2020-10-04T02:30:00 does not exist: the clocks are moved forward one hour for DST on that day at 02:00. However, such a value may appear during calculations (see duration semantics in Section 1.4.6), or due to a change in time zone rules (so it was valid when the event was first created). Again, it is interpreted as though the offset before

the transition is in effect (+10:00), therefore when converted to UTC we get 2020-10-03T16:30:00Z.

1.4.6. Duration

Where Duration is given as a type, it means a length of time represented by a subset of ISO8601 duration format, as specified by the following ABNF [RFC5234]:

```
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"
dur-cal     = (dur-week [dur-day] / dur-day)

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

In addition, the duration MUST NOT include fractional second values unless the fraction is non-zero. Fractional second values MUST NOT have trailing zeros, to ensure there is only a single representation for each duration.

A duration specifies an abstract number of weeks, days, hours, minutes, and/or seconds. A duration specified using weeks or days does not always correspond to an exact multiple of 24 hours. The number of 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 adding or subtracting a duration to/from a LocalDateTime.

To add a duration to a LocalDateTime:

1. Add any week or day components of the duration to the date. A week is always the same as 7 days.

2. If a time zone applies to the LocalDateTime, convert it to a UTCDateTime following the semantics in Section 1.4.5.

3. Add any hour, minute or second components of the duration (in absolute time).

4. Convert the resulting UTCDateTime back to a LocalDateTime in the time zone that applies.

To subtract a duration from a LocalDateTime, the steps apply in reverse:

1. If a time zone applies to the LocalDateTime, convert it to UTC following the semantics in Section 1.4.5.

2. Subtract any hour, minute or second components of the duration (in absolute time).

3. Convert the resulting UTCDateTime back to LocalDateTime in the time zone that applies.

4. Subtract any week or day components of the duration from the date.

5. If the resulting time does not exist on the date due to a discontinuity in the time zone, use the semantics in Section 1.4.5 to convert to UTC and back to get a valid LocalDateTime.

These semantics match the iCalendar DURATION value type ([RFC5545], Section 3.3.6).

1.4.7. SignedDuration

A SignedDuration represents a length of time that may be positive or negative and is typically used to express the offset of a point in time relative to an associated time. It is represented as a Duration, optionally preceded by a sign character. 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.

1.4.8. TimeZoneId

Where "TimeZoneId" is given as a data type, it means a "String" that is either a time zone name in the IANA Time Zone Database [TZDB] or a custom time zone identifier defined in the "timeZones" property (see Section 4.7.2).

Where an IANA time zone is specified, the zone rules of the respective zone records apply. Custom time zones are interpreted as described in Section 4.7.2.

1.4.9. PatchObject

A PatchObject is of type "String[*]", and represents an unordered set of patches on a JSON object. Each key is a path represented in a subset of JSON pointer format [RFC6901]. The paths have an implicit leading "/", so each key is prefixed 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., you MUST NOT insert/delete from an array; the array MUST be replaced in its entirety instead).

2. All parts prior to the last (i.e., the value after the final slash) MUST already exist on the object being patched.

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

4. The value for the patch MUST be valid for the property being set (of the correct type and obeying any other applicable restrictions), or if null the property MUST be optional.

The value associated with each pointer determines how to apply that patch:

- If null, remove the property from the patched object. If the key is not present in the parent, this a no-op.

- Anything else: The value to set for this property (this may be a replacement or addition to the object being patched).

A PatchObject does not define its own "@type" property (see Section 4.1.1). A "@type" property in a patch MUST be handled as any other patched property value.

Implementations MUST reject in its entirety a PatchObject if any of its patches is invalid. Implementations MUST NOT apply partial patches.

The PatchObject format is used to significantly reduce file size and duplicated content when specifying variations to a common object, such as with recurring events or when translating the data into multiple languages. It can also better preserve semantic intent if only the properties that should differ between the two objects are

patched. For example, if one person is not going to a particular instance of a regularly scheduled event, in iCalendar you would have to duplicate the entire event in the override. In JSCalendar this is a small patch to show the difference. As only this property is patched, if the location of the event is changed, the occurrence will automatically still inherit this.

1.4.10. Relation

A Relation object defines the relation to other objects, using a possibly empty set of relation types. The object that defines this relation is the linking object, while the other object is the linked object. A Relation object has the following properties:

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "Relation".

- relation: "String[Boolean]" (optional, default: empty Object)

  Describes how the linked object is related to the linking object. The relation is defined as a set of relation types. If empty, the relationship between the two objects is unspecified.

  Keys in the set MUST be one of the following values, or specified in the property definition where the Relation object is used, or a value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

  * "first": The linked object is the first in a series the linking object is part of.

  * "next": The linked object is the next in a series the linking object is part of.

  * "child": The linked object is a subpart of the linking object.

  * "parent": The linking object is a subpart of the linked object.

  The value for each key in the map MUST be true.

1.4.11. Link

A Link object represents an external resource associated with the linking object. It has the following properties:

- @type: "String" (mandatory)

Specifies the type of this object. This MUST be "Link".

- href: "String" (mandatory)

  A URI [RFC3986] from which the resource may be fetched.

  This MAY be a "data:" URL [RFC2397], but it is recommended that the file be hosted on a server to avoid embedding arbitrarily large data in JSCalendar object instances.

- cid: "String" (optional)

  This MUST be a valid "content-id" value according to the definition of Section 2 in [RFC2392]. The value MUST be unique within this Link object but has no meaning beyond that. It MAY be different from the link id for this Link object.

- contentType: "String" (optional)

  The media type [RFC6838] of the resource, if known.

- size: "UnsignedInt" (optional)

  The size, in octets, of the resource when fully decoded (i.e., the number of octets in the file the user would download), if known. Note that this is an informational estimate, and implementations must be prepared to handle the actual size being quite different when the resource is fetched.

- rel: "String" (optional)

  Identifies the relation of the linked resource to the object. If set, the value MUST be a relation type from the IANA registry [LINKRELS], as established in [RFC8288].

- 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 one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

  * "badge": an image meant to be displayed alongside the title of the object.

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

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

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

- title: "String" (optional)

  A human-readable plain-text description of the resource.

2. JSCalendar Objects

This section describes the calendar object types specified by JSCalendar.

2.1. JSEvent

Media type: "application/jscalendar+json;type=jsevent"

A JSEvent represents a scheduled amount of time on a calendar, typically a meeting, appointment, reminder or anniversary. It is required to start at a certain point in time and typically has a non-zero duration. Multiple participants may partake in the event at multiple locations.

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

2.2. JSTask

Media type: "application/jscalendar+json;type=jstask"

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

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

2.3. JSGroup

Media type: "application/jscalendar+json;type=jsgroup"

A JSGroup is a collection of JSEvent (Section 2.1) and/or 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 [RFC8259], which MUST be valid I-JSON (a stricter subset of JSON) [RFC7493]. 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. Object Type

JSCalendar objects MUST name their type in the "@type" property, if not explicitly specified otherwise for the respective object type. A notable exception to this rule is the PatchObject (Section 1.4.9).

3.2. 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.

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

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

- Several JSCalendar property values are defined as URIs and media 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 elsewhere.

3.3. Vendor-specific Property Extensions, Values and Types

Vendors MAY add additional properties to the calendar object to support their custom features. To avoid conflict, the names of these properties MUST be prefixed by a domain name controlled by the vendor followed by a colon, e.g., "example.com:customprop". If the value is

a new JSCalendar object, it either MUST include a "@type" property or it MUST explicitly be specified to not require a type designator. The type name MUST be prefixed with a domain name controlled by the vendor.

Some JSCalendar properties allow vendor-specific value extensions. Such vendor-specific values MUST be prefixed by a domain name controlled by the vendor followed by a colon, e.g., "example.com:customrel".

Vendors are strongly encouraged to register any new property values or extensions that are useful to other systems as well, rather than use a vendor-specific prefix.

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:

- "jsevent": a JSCalendar event (Section 2.1).
- "jstask": a JSCalendar task (Section 2.2).
- "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). UUID version 4, described in Section 4.4 of [RFC4122], 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.

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 occurrences 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 last updated the JSCalendar object. This should be set whenever the data in the object is modified (i.e., whenever the "updated" property is set).

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

This property SHOULD NOT be used to alter the interpretation of a 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, a practice that is known to cause long-term interoperability problems.

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 (or its creation date/time if not modified since).

4.1.7. sequence

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

Initially zero, this MUST be incremented by one every time a change is made to the object, except if the change only modifies the "participants" property (see Section 4.4.5).

This is used as part of iTIP [RFC5546] to know which version of the object a scheduling message relates to.

4.1.8. method

Type: "String" (optional).

The iTIP [RFC5546] method, in lowercase. This MUST only be present if the JSCalendar object represents an iTIP scheduling message.

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" [MEDIATYPES]. They MAY include 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 that the time is not important to display to the user when rendering this calendar object. An example of this is 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.

Such events are also commonly known as "all-day" events.

4.2.5. locations

Type: "Id[Location]" (optional).

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

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

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "Location".

- name: "String" (optional)

  The human-readable name of the location.

- description: "String" (optional)

  Human-readable, plain-text instructions for accessing this location. This may be an address, set of directions, door access code, etc.

- locationTypes: "String[Boolean]" (optional)

  A set of one or more location types that describe this location. All types MUST be from the Location Types Registry [LOCATIONTYPES] as defined in [RFC4589]. The set is represented as a map, with

the keys being the location types. The value for each key in the map MUST be true.

- relativeTo: "String" (optional)

  Specifies the relation between this location and the time of the JSCalendar object. This is primarily to allow events representing travel to specify the location of departure (at the start of the event) and location of arrival (at the end); this is particularly important if these locations are in different time zones, as a client may wish to highlight this information for the user.

  This MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3). Any value the client or server doesn't understand should be treated the same as if this property is omitted.

  * "start": The event/task described by this JSCalendar object occurs at this location at the time the event/task starts.

  * "end": The event/task described by this JSCalendar object occurs at this location at the time the event/task ends.

- timeZone: "TimeZoneId" (optional)

  A time zone for this location.

- coordinates: "String" (optional)

  A "geo:" URI [RFC5870] for the location.

- links: "Id[Link]" (optional)

  A map of link ids to Link objects, representing external resources associated with this location, for example a vCard or image. If there are no links, this MUST be omitted (rather than specified as an empty set).

4.2.6. virtualLocations

Type: "Id[VirtualLocation]" (optional).

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

A VirtualLocation object has the following properties.

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "VirtualLocation".

- name: "String" (optional, default: empty String)

  The human-readable name of the virtual location.

- description: "String" (optional)

  Human-readable plain-text instructions for accessing this virtual location. This may be a conference access code, etc.

- uri: "String" (mandatory)

  A URI [RFC3986] that represents how to connect to this virtual location.

  This may be a telephone number (represented using the "tel:" scheme, e.g., "tel:+1-555-555-5555") for a teleconference, a web address for online chat, or any custom URI.

4.2.7. links

Type: "Id[Link]" (optional).

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

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

Links with a rel of "describedby" MUST be considered by the client to be an alternative representation of the description.

Links with a rel of "icon" MUST 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.

4.2.8. locale

Type: "String" (optional).

The language tag as defined in [RFC5646] that best describes the locale used for the text in 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).

A color clients MAY use when displaying this calendar object. The value is a color name taken from the set of names defined in Section 4.3 of CSS Color Module Level 3 [COLORS], or an RGB value in hexadecimal notation, as defined in Section 4.2.1 of CSS Color Module Level 3.

4.3. Recurrence Properties

Some events and tasks occur at regular or irregular intervals. Rather than having to copy the data for every occurrence there can be a master event with rules to generate recurrences, and/or overrides that add extra dates or exceptions to the rules.

The recurrence set is the complete set of instances for an object. It is generated by considering the following properties in order, all of which are optional:

1. The recurrenceRules property (Section 4.3.2) generates a set of extra date-times on which the object occurs.

2. The excludedRecurrenceRules property (Section 4.3.3) generates a set of date-times that are to be removed from the previously generated set of date-times on which the object occurs.

3. The recurrenceOverrides property (Section 4.3.4) defines date-times which are added or excluded to form the final set. (This property may also contain changes to the object to apply to particular instances.)

4.3.1. recurrenceId

Type: "LocalDateTime" (optional).

If present, this JSCalendar object represents one occurrence of a recurring JSCalendar object. If present the "recurrenceRules" and "recurrenceOverrides" properties MUST NOT be present.

The value is a date-time either produced by the "recurrenceRules" of the master event, or added as a key to the "recurrenceOverrides" property of the master event.

4.3.2. recurrenceRules

Type: "RecurrenceRule[]" (optional).

Defines a set of recurrence rules (repeating patterns) for recurring calendar objects.

A JSEvent recurs by applying the recurrence rules to the "start" date-time.

A JSTask recurs by applying the recurrence rules to the "start" date-time, if defined, otherwise it recurs by the "due" date-time, if defined. If the task defines neither a "start" nor "due" date-time, it MUST NOT define a "recurrenceRules" property.

If multiple recurrence rules are given, each rule is to be applied and then the union of the results used, ignoring any duplicates.

A RecurrenceRule object is a JSON object mapping of a RECUR value type in iCalendar [RFC5545] [RFC7529] and has the same semantics. It has the following properties:

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "RecurrenceRule".

- frequency: "String" (mandatory)

  The time span covered by each iteration of this recurrence rule (see Section 4.3.2.1 for full semantics). This MUST be one of the following values:

  * "yearly"

  * "monthly"

  * "weekly"

  * "daily"

  * "hourly"

  * "minutely"

  * "secondly"

  This is the FREQ part from iCalendar, converted to lowercase.

- interval: "UnsignedInt" (optional, default: 1)

  The interval of iteration periods at which the recurrence repeats. If included, it MUST be an integer >= 1.

  This is the INTERVAL part from iCalendar.

- rscale: "String" (optional, default: "gregorian")

  The calendar system in which this recurrence rule operates, in lowercase. This MUST be either a CLDR-registered calendar system name [CLDR], or a vendor-specific value (see Section 3.3).

  This is the RSCALE part from iCalendar RSCALE [RFC7529], converted to lowercase.

- skip: "String" (optional, default: "omit")

  The behaviour to use when the expansion of the recurrence produces invalid dates. This property only has an effect if the frequency is "yearly" or "monthly". It MUST be one of the following values:

  * "omit"

  * "backward"

  * "forward"

  This is the SKIP part from iCalendar RSCALE [RFC7529], converted to lowercase.

- firstDayOfWeek: "String" (optional, default: "mo")

The day on which the week is considered to start, represented as a lowercase 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"

This is the WKST part from iCalendar.

- byDay: "NDay[]" (optional)

  Days of the week on which to repeat. An "NDay" object has the following properties:

  * @type: "String" (mandatory)

    Specifies the type of this object. This MUST be "NDay".

  * day: "String" (mandatory)

    A day of the week on which to repeat; the allowed values are the same as for the "firstDayOfWeek" RecurrenceRule property.

    This is the day-of-the-week of the BYDAY part in iCalendar, converted to lowercase.

  * nthOfPeriod: "Int" (optional)

    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.

    This is the ordinal part of the BYDAY value in iCalendar (e.g., 1 or -3).

- byMonthDay: "Int[]" (optional)

  Days of the month on which to repeat. Valid values are between 1 and the maximum number of days any month may have in the calendar given by the "rscale" property, and the negative values of these numbers. For example, in the Gregorian calendar valid values are 1 to 31 and -31 to -1. Negative values offset from the end of the month. The array MUST have at least one entry if included.

  This is the BYMONTHDAY part in iCalendar.

- byMonth: "String[]" (optional)

  The months in which to repeat. 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 the Gregorian calendar), with an optional "L" suffix (see [RFC7529]) for leap months (this MUST be uppercase, e.g., "3L"). The array MUST have at least one entry if included.

  This is the BYMONTH part from iCalendar.

- byYearDay: "Int[]" (optional)

  The days of the year on which to repeat. Valid values are between 1 and the maximum number of days any year may have in the calendar given by the "rscale" property, and the negative values of these numbers. For example, in the Gregorian calendar valid values are 1 to 366 and -366 to -1. Negative values offset from the end of the year. The array MUST have at least one entry if included.

  This is the BYYEARDAY part from iCalendar.

- byWeekNo: "Int[]" (optional)

  Weeks of the year in which to repeat. Valid values are between 1 and the maximum number of weeks any year may have in the calendar given by the "rscale" property, and the negative values of these numbers. For example, in the Gregorian calendar valid values are 1 to 53 and -53 to -1. The array MUST have at least one entry if included.

  This is the BYWEEKNO part from iCalendar.

- byHour: "UnsignedInt[]" (optional)

The hours of the day in which to repeat. Valid values are 0 to 23. The array MUST have at least one entry if included. This is the BYHOUR part from iCalendar.

- byMinute: "UnsignedInt[]" (optional)

  The minutes of the hour in which to repeat. Valid values are 0 to 59. The array MUST have at least one entry if included.

  This is the BYMINUTE part from iCalendar.

- bySecond: "UnsignedInt[]" (optional)

  The seconds of the minute in which to repeat. Valid values are 0 to 60. The array MUST have at least one entry if included.

  This is the BYSECOND part from iCalendar.

- bySetPosition: "Int[]" (optional)

  The occurrences within the recurrence interval to include in the final results. Negative values offset from the end of the list of occurrences. The array MUST have at least one entry if included. This is the BYSETPOS part from iCalendar.

- count: "UnsignedInt" (optional)

  The number of occurrences at which to range-bound the recurrence. This MUST NOT be included if an "until" property is specified.

  This is the COUNT part from iCalendar.

- until: "LocalDateTime" (optional)

  The date-time at which to finish recurring. The last occurrence is on or before this date-time. This MUST NOT be included if a "count" property is specified. Note: if not specified otherwise for a specific JSCalendar object, this date is to be interpreted in the time zone specified in the JSCalendar object's "timeZone" property.

  This is the UNTIL part from iCalendar.

4.3.2.1. Interpreting recurrence rules

A recurrence rule specifies a 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, the date-time 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 passing 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 is "forward") or the last day of the current month (if skip is "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 is "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 initial date-time to which the rule is applied (the "start" date-time for events; the "start" or "due" date-time for tasks) 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 initial date-time. This means the first set may include candidates before the initial date-time; 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 is not "secondly" and no bySecond property: Add a bySecond property with the sole value being the seconds value of the initial date-time.

* If frequency is not "secondly" or "minutely", and no byMinute property: Add a byMinute property with the sole value being the minutes value of the initial date-time.

* If frequency is not "secondly", "minutely" or "hourly" and no byHour property: Add a byHour property with the sole value being the hours value of the initial 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 initial 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 initial 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 initial 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 initial 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 initial date-time.

4.3.3. excludedRecurrenceRules

Type: "RecurrenceRule[]" (optional).

Defines a set of recurrence rules (repeating patterns) for date-times on which the object will not occur. The rules are interpreted the same as for the "recurrenceRules" property (see Section 4.3.2), with the exception that the initial date-time to which the rule is applied (the "start" date-time for events; the "start" or "due" date-time for tasks) is only considered part of the expansion if it matches the rule. The resulting set of date-times are then removed from those generated by the recurrenceRules property, as described in Section 4.3.

4.3.4. recurrenceOverrides

Type: "LocalDateTime[PatchObject]" (optional).

A map of the recurrence ids (the date-time produced by the recurrence rule) to an object of patches to apply to the generated occurrence object.

If the recurrence id does not match a date-time from the recurrence rule (or no rule is specified), 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 of an occurrence to be true, this occurrence is omitted from the final set of recurrences for the calendar object (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 match the recurrence id LocalDateTime. 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 value generated from the recurrence id. Both the recurrence id 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:

- @type
- excludedRecurrenceRules
- method
- privacy
- prodId
- recurrenceId
- recurrenceOverrides
- recurrenceRules
- relatedTo

- replyTo
- uid

4.3.5. excluded

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

Defines if this object is an overridden, excluded instance of a recurring JSCalendar object (see Section 4.3.4). If this property value is true, this calendar object instance MUST be removed from the occurrence expansion. The absence of this property, or the presence of its default value false, indicates that this instance MUST be included in the occurrence expansion.

4.4. Sharing and Scheduling Properties

4.4.1. priority

Type: "Int" (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, for which the treatment will vary by situation. 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 calendar object should be treated when calculating free-busy state. This MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

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

4.4.3. privacy

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

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

This property MUST NOT affect the information sent to scheduled participants; it is only interpreted by protocols that share the calendar objects belonging to one user with other users.

The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3). Any value the client or server doesn't understand should be preserved but treated as equivalent to "private".

- "public": The full details of the object are visible to those whom the object's calendar is shared with.
- "private": The details of the object are hidden; only the basic time and metadata is shared. The following properties MAY be shared, any other properties MUST NOT be shared:
  - @type
  - created
  - due
  - duration
  - estimatedDuration
  - freeBusyStatus
  - privacy
  - recurrenceOverrides. Only patches which apply to another permissible property are allowed to be shared.
  - sequence
  - showWithoutTime

  * start
  * timeZone
  * timeZones
  * uid
  * updated

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

4.4.4. replyTo

Type: "String[String]" (optional).

Represents methods by which participants may submit their 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 for the method specified in the key. Future methods may be defined in future specifications and registered with IANA; 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 being specified as an empty object).

The following methods are defined:

- "imip": The organizer accepts an iMIP [RFC6047] response at this email address. The value MUST be a "mailto:" URI.
- "web": Opening this URI in a web browser will provide the user with a page where they can submit a reply to the organizer. The value MUST be a URL using the "https:" scheme.
- "other": The organizer is identified by this URI but the method for submitting the response is undefined.

4.4.5. participants

Type: "Id[Participant]" (optional).

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

If this property is set and any participant has a sendTo property, then the "replyTo" property of this calendar object MUST define at least one reply method.

A Participant object has the following properties:

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "Participant".

- name: "String" (optional)

  The display name of the participant (e.g., "Joe Bloggs").

- email: "String" (optional)

  The email address for the participant.

- description: "String" (optional).

  A plain text description of this participant. For example, this may include more information about their role in the event or how best to contact them.

- sendTo: "String[String]" (optional)

  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 for the method specified in the key. Future methods may be defined in future specifications and registered with IANA; 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 being specified as 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 for submitting the invitation is undefined.

- kind: "String" (optional)

  What kind of entity this participant is, if known.

  This MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3). 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

  * "location": a physical location that needs to be scheduled, e.g., a conference room

  * "resource": a non-human resource other than a location, such as a projector

- roles: "String[Boolean]" (mandatory)

  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 one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

  * "owner": The participant is an owner of the object. This signifies they have permission to make changes to it that affect the other participants. Non-owner participants may only change properties that just affect themselves (for example setting their own alerts or changing their rsvp status).

  * "attendee": The participant is expected to be present at the event.

  * "optional": The participant's involvement with the event is optional. This is expected to be primarily combined with the "attendee" role.

  * "informational": The participant is copied for informational reasons, and is not expected to attend.

  * "chair": The participant is in charge of the event/task when it occurs.

* "contact": The participant is someone that may be contacted for information about the event.

The value for each key in the map MUST be true. It is expected that no more than one of the roles "attendee" and "informational" be present; if more than one are given, "attendee" takes precedence over "informational". Roles that are unknown to the implementation MUST be preserved.

- 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 JSCalendar object, this MUST be treated the same as if the participant's locationId were omitted.

- language: "String" (optional)

  The language tag as defined in [RFC5646] that best describes the participant's preferred language, if known.

- participationStatus: "String" (optional, default: "needs-action")

  The participation status, if any, of this participant.

  The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

  * "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.

  * "delegated": The invited participant has delegated their attendance to another participant, as specified in the delegatedTo property.

- participationComment: "String" (optional)

  A note from the participant to explain their participation status.

- expectReply: "Boolean" (optional, default: false)

If true, the organizer is expecting the participant to notify them of their participation status.

- scheduleAgent: "String" (optional, default: "server")

  Who is responsible for sending scheduling messages with this calendar object to the participant.

  The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

  * "server": The calendar server will send the scheduling messages.

  * "client": The calendar client will send the scheduling messages.

  * "none": No scheduling messages are to be sent to this participant.

- scheduleForceSend: "Boolean" (optional, default: false)

  A client may set the property on a participant to true to request that the server send a scheduling message to the participant when it would not normally do so (e.g. if no significant change is made the object or the scheduleAgent is set to client). The property MUST NOT be stored in the JSCalendar object on the server or appear in a scheduling message.

- scheduleSequence: "UnsignedInt" (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 response following significant changes to the calendar object, and to determine if future responses are responding to a current or older view of the data.

- scheduleStatus: "String[]" (optional)

  A list of status codes, as defined in Section 3.8.8.3 of [RFC5545], returned from the processing of the most recent scheduling message sent to this participant.

  Servers MUST only add or change this property when they send a scheduling message to the participant. Clients SHOULD NOT change

or remove this property if it was provided by the server. Clients MAY add, change, or remove the property for participants where the client is handling the scheduling.

This property MUST NOT be included in scheduling messages.

- scheduleUpdated: "UTCDateTime" (optional)

  The timestamp for the most recent response from this participant.

  This is the "updated" property of the last response when using iTIP. It can be compared to the "updated" property in future responses to detect and discard older responses delivered out of order.

- invitedBy: "String" (optional)

  The participant id of the participant who invited this one, if known.

- delegatedTo: "String[Boolean]" (optional)

  A set of participant ids that this participant has delegated their participation to. Each key in the set MUST be the id of a participant. The value for each key in the map MUST be true. If there are no delegates, this MUST be omitted (rather than specified as an empty set).

- delegatedFrom: "String[Boolean]" (optional)

  A set of participant ids that this participant is acting as a delegate for. Each key in the set MUST be the id of a participant. The value for each key in the map MUST be true. If there are no delegators, this MUST be omitted (rather than specified as an empty set).

- memberOf: "String[Boolean]" (optional)

  A set of group participants that were invited to this calendar object, which caused this participant to be invited due to their membership in the group(s). Each key in the set MUST be the id of a participant. The value for each key in the map MUST be true. If there are no groups, this MUST be omitted (rather than specified as an empty set).

- links: "Id[Link]" (optional)

A map of link ids to Link objects, representing external resources associated with this participant, for example a vCard or image. If there are no links, this MUST be omitted (rather than specified as an empty set).

- progress: "String" (optional; only allowed for participants of a JSTask). Represents the progress of the participant for this task. It MUST NOT be set if the "participationStatus" of this participant is any value other than "accepted". See Section 5.2.5 for allowed values and semantics.

- progressUpdated: "UTCDateTime" (optional; only allowed for participants of a JSTask). Specifies the date-time the progress property was last set on this participant. See Section 5.2.6 for allowed values and semantics.

- percentComplete: "UnsignedInt" (optional; only allowed for participants of a JSTask). Represents the percent completion of the participant for this task. The property value MUST be a positive integer between 0 and 100.

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: "Id[Alert]" (optional).

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

An Alert Object has the following properties:

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "Alert".

- trigger: "OffsetTrigger|AbsoluteTrigger|UnknownTrigger" (mandatory)

  Defines when to trigger the alert. New types may be defined in future documents.

  An "OffsetTrigger" object has the following properties:

  * @type: "String" (mandatory)

    Specifies the type of this object. This MUST be "OffsetTrigger".

  * offset: "SignedDuration" (mandatory).

    Defines the offset at which to trigger the alert relative to the time property defined in the "relativeTo" property of the alert. Negative durations signify alerts before the time property, positive durations signify alerts after.

  * relativeTo: "String" (optional, default: "start")

    Specifies the time property that 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 "AbsoluteTrigger" object has the following properties:

  * @type: "String" (mandatory)

    Specifies the type of this object. This MUST be "AbsoluteTrigger".

  * when: "UTCDateTime" (mandatory).

    Defines a specific UTC date-time when the alert is triggered.

  An "UnknownTrigger" object is an object that contains a "@type" property whose value is not recognized (i.e., not "OffsetTrigger" or "AbsoluteTrigger"), plus zero or more other properties. This is for compatibility with client extensions and future specifications. Implementations SHOULD NOT trigger for trigger types they do not understand, but MUST preserve them.

- acknowledged: "UTCDateTime" (optional)

  This records when an alert was last acknowledged. This is set when the user has dismissed the alert; other clients that sync this property SHOULD 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, setting the "acknowledged" property MUST NOT add a new override to the "recurrenceOverrides" property. If the alert is not already overridden, the acknowledged property MUST be set on the alert in the master event/task.

  Certain kinds of alert action may not provide feedback as to when the user sees them, for example email based alerts. For those kinds of alerts, this property MUST be set immediately when the alert is triggered and the action successfully carried out.

- relatedTo: "String[Relation]" (optional)

  Relates this alert to other alerts in the same JSCalendar object. If the user wishes to snooze an alert, the application MUST create an alert to trigger after snoozing. This new snooze alert MUST set a parent relation to the identifier of the original alert.

- action: "String" (optional, default: "display")

  Describes how to alert the user.

  The value MUST be at most one of the following values, a value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

  * "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 language tags [RFC5646] to patch objects, which localize the calendar object into the locale of the respective language tag.

See the description of PatchObject (Section 1.4.9) for the structure of the PatchObject. The patches are applied to the top-level calendar object. In addition, the "locale" property of the patched object is set to the language tag. All pointers for patches MUST end with one of the following suffixes; any patch that does not follow this MUST be ignored unless otherwise specified in a future RFC:

- title
- description
- name

A patch MUST NOT have the prefix "recurrenceOverrides"; any localization of the override MUST be a patch to the localizations property inside the override instead. For example, a patch to "locations/abcd1234/title" is permissible, but a patch to "uid" or "recurrenceOverrides/2020-01-05T14:00:00/title" is not.

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. timeZone

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

Identifies the time zone the object is scheduled in, or null for floating time. This is either a name from the IANA Time Zone Database [TZDB] or the TimeZoneId of a custom time zone from the "timeZones" property (Section 4.7.2). If omitted, this MUST be presumed to be null (i.e., floating time).

4.7.2. timeZones

Type: "TimeZoneId[TimeZone]" (optional).

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

- To avoid conflict with names in the IANA Time Zone Database [TZDB], it MUST start with the "/" character.

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

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

An identifier need only be unique to this JSCalendar object. A JSCalendar object may be part in a hierarchy of other JSCalendar objects (say, a JSEvent is an entry in a JSGroup). In this case, the set of time zones is the sum of the time zone definitions of this object and its parent objects. If multiple time zones with the same identifier exist, then the definition closest to the calendar object in relation to its parents MUST be used. (In context of JSEvent, a time zone definition in its timeZones property has precedence over a definition of the same id in the JSGroup). Time zone definitions in any children of the calendar object MUST be ignored.

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

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "TimeZone".

- tzId: "String" (mandatory).

  The TZID property from iCalendar.

- updated: "UTCDateTime" (optional)

  The LAST-MODIFIED property from iCalendar.

- url: "String" (optional)

  The TZURL property from iCalendar.

- validUntil: "UTCDateTime" (optional)

  The TZUNTIL property from iCalendar specified in [RFC7808].

- aliases: "String[Boolean]" (optional)

Maps the TZID-ALIAS-OF properties from iCalendar specified in [RFC7808] to a JSON set of aliases. The set is represented as an object, with the keys being the aliases. The value for each key in the map MUST be true.

- standard: "TimeZoneRule[]" (optional)

  The STANDARD sub-components from iCalendar. The order MUST be preserved during conversion.

- daylight: "TimeZoneRule[]" (optional).

  The DAYLIGHT sub-components from iCalendar. The order MUST be preserved during conversion.

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

- @type: "String" (mandatory)

  Specifies the type of this object. This MUST be "TimeZoneRule".

- start: "LocalDateTime" (mandatory)

  The DTSTART property from iCalendar.

- offsetFrom: "String" (mandatory)

  The TZOFFSETFROM property from iCalendar.

- offsetTo: "String" (mandatory)

  The TZOFFSETTO property from iCalendar.

- recurrenceRules: "RecurrenceRule[]" (optional)

  The RRULE property mapped as specified in Section 4.3.2. During recurrence rule evaluation, the "until" property value MUST be interpreted as a local time in the UTC time zone.

- recurrenceOverrides: "LocalDateTime[PatchObject]" (optional)

  Maps the RDATE properties from iCalendar. The set is represented as an object, with the keys being the recurrence dates. The patch object MUST be the empty JSON object ({}).

- 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, excluding any "tznparam" component from iCalendar. The value for each key in the map MUST be true.

- comments: "String[]" (optional). Maps the COMMENT properties from iCalendar. The order MUST be preserved during conversion.

5. Type-specific JSCalendar Properties

5.1. JSEvent Properties

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

5.1.1. start

Type: "LocalDateTime" (mandatory).

The date/time the event starts in the event's time zone (as specified in the "timeZone" property, see Section 4.7.1).

5.1.2. duration

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

The zero or positive duration of the event in the event's start time zone. The end time of an event can be found by adding the duration to the event's start time.

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 (see Section 4.2.5).

5.1.3. status

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

The scheduling status (Section 4.4) of a JSEvent. If set, it MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- "confirmed": Indicates the event is definitely happening.

- "cancelled": Indicates the event has been cancelled.

- "tentative": Indicates the event may happen.

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. estimatedDuration

Type: "Duration" (optional).

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

5.2.4. percentComplete

Type: "UnsignedInt" (optional).

Represents the percent completion of the task overall. The property value MUST be a positive integer between 0 and 100.

5.2.5. progress

Type: "String" (optional).

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

- "completed": if the "progress" property value of all participants is "completed".
- "failed": if at least one "progress" property value of a participant is "failed".

- "in-process": if at least one "progress" property value of a participant is "in-process".

- "needs-action": If none of the other criteria match.

If set, it MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- "needs-action": Indicates the task needs action.

- "in-process": Indicates the task is in process.

- "completed": Indicates the task is completed.

- "failed": Indicates the task failed.

- "cancelled": Indicates the task was cancelled.

5.2.6. progressUpdated

Type: "UTCDateTime" (optional).

Specifies the date/time the "progress" property of either the task overall (Section 5.2.5) or a specific participant (Section 4.4.5) was last updated.

If the task is recurring and has future instances, a client may want to keep track of the last progress update timestamp of a specific task recurrence, but leave other instances unchanged. One way to achieve this is by overriding the progressUpdated 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 progress update time and a future recurring instance. See also Section 4.1.3 on splitting.

5.3. JSGroup Properties

JSGroup supports the following common JSCalendar properties (Section 4):

- @type

- uid

- prodId

- created
- updated
- title
- description
- descriptionContentType
- links
- locale
- keywords
- categories
- color
- timeZones

In addition, the following JSGroup-specific properties are supported:

5.3.1. entries

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

A collection of group members. 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. 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, 2020 at 1pm New York local time and ends after 1 hour.

```
{
  "@type": "jsevent",
  "uid": "a8df6573-0474-496d-8496-033ad45d7fea",
  "updated": "2020-01-02T18:23:04Z",
  "title": "Some event",
  "start": "2020-01-15T13:00:00",
  "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": "2020-01-09T14:32:01Z",
  "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": "bf0ac22b-4989-4caf-9ebd-54301b4ee51a",
  "updated": "2020-01-15T18:00:00Z",
  "name": "A simple group",
  "entries": [{
    "@type": "jsevent",
    "uid": "a8df6573-0474-496d-8496-033ad45d7fea",
    "updated": "2020-01-02T18:23:04Z",
    "title": "Some event",
    "start": "2020-01-15T13:00:00",
    "timeZone": "America/New_York",
    "duration": "PT1H"
  },
  {
    "@type": "jstask",
    "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
    "updated": "2020-01-09T14:32:01Z",
    "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",
  "recurrenceRules": [{
    "@type": "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, 2020. The calendar user expects to need 1 hour for shopping.

```
{
  "...": "",
  "title": "Buy groceries",
  "due": "2020-01-19T18:00:00",
  "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, 2020 at 9am in Berlin local time. The duration of the flight is scheduled at 10 hours 30 minutes. The time at the flight's 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": "2020-04-01T09:00:00",
  "timeZone": "Europe/Berlin",
  "duration": "PT10H30M",
  "locations": {
    "418d0b9b-b656-4b3c-909f-5b149ca779c9": {
      "@type": "Location",
      "rel": "start",
      "name": "Frankfurt Airport (FRA)"
    },
    "c2c7ac67-dc13-411e-a7d4-0780fb61fb08": {
      "@type": "Location",
      "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, 2020, 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": "2020-01-01T07:00:00",
     "duration": "PT30M",
     "recurrenceRules": [{
       "@type": "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": "2020-07-04T17:00:00",
  "timeZone": "America/New_York",
  "duration": "PT3H",
  "locations": {
    "c0503d30-8c50-4372-87b5-7657e8e0fedd": {
      "@type": "Location",
      "name": "The Music Bowl",
      "description": "Music Bowl, Central Park, New York",
      "coordinates": "geo:40.7829,-73.9654"
    }
  },
  "virtualLocations": {
    "1": {
      "@type": "VirtualLocation",
      "name": "Free live Stream from Music Bowl",
      "uri": "https://stream.example.com/the_band_2020"
    }
  },
  "localizations": {
    "de": {
      "title": "Live von der Music Bowl: The Band!",
      "description": "Schau dir das groesste Musikereignis an!",
      "virtualLocations/1/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, 2020 at 9am London time and occurs every week until June 24, 2020. 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 1 no course is held. On January 7 at 2pm is an optional introduction course, that occurs before the first regular lecture.

```
{
  "...": "",
  "title": "Calculus I",
  "start": "2020-01-08T09:00:00",
  "timeZone": "Europe/London",
  "duration": "PT1H30M",
  "locations": {
    "0dfb8ace-aad1-4734-b3b4-a2fe3d6ae1c5": {
      "@type": "Location",
      "title": "Math lab room 1",
      "description": "Math Lab I, Department of Mathematics"
    }
  },
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "weekly",
    "until": "2020-06-24T09:00:00"
  }],
  "recurrenceOverrides": {
    "2020-01-07T14:00:00": {
      "title": "Introduction to Calculus I (optional)"
    },
    "2020-04-01T09:00:00": {
      "excluded": true
    },
    "2020-06-25T09:00:00": {
      "title": "Calculus I Exam",
      "start": "2020-06-25T10:00:00",
      "duration": "PT2H",
      "locations": {
        "84d639ca-37ac-4a86-81e5-9bbba8eb4053": {
          "@type": "Location",
          "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, 2020 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 4, 2020 he is unavailable and declined participation for this occurrence.

```
{
  "...": "",
  "title": "FooBar team meeting",
  "start": "2020-01-08T09:00:00",
  "timeZone": "Africa/Johannesburg",
  "duration": "PT1H",
  "virtualLocations": {
    "3f41b47b-a5eb-494f-90eb-19d279486d84": {
      "@type": "VirtualLocation",
      "name": "ChatMe meeting room",
      "uri": "https://chatme.example.com?id=1234567&pw=a8a24627b63d396e"
    }
  },
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "weekly"
  }],
  "replyTo": {
    "imip": "mailto:f245f875-7f63-4a5e-a2c8@schedule.example.com"
  },
  "participants": {
    "dG9tQGZvb2Jhci5xlLmNvbQ": {
      "@type": "Participant",
      "name": "Tom Tool",
      "email": "tom@foobar.example.com",
      "sendTo": {
        "imip": "mailto:tom@calendar.example.com"
      },
      "participationStatus": "accepted",
      "roles": {
        "attendee": true
      }
    },
    "em9lQGZvb2GFtcGxlLmNvbQ": {
      "@type": "Participant",
      "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": {
    "2020-03-04T09:00:00": {
      "participants/dG9tQGZvb2Jhci5xlLmNvbQ/participationStatus":
                                                              "declined"
    }
  }
}
```

7. Security Considerations

Calendaring and scheduling information is very privacy-sensitive. It can reveal the social network of a user; location information of this user and those in their social network; identity and credentials information; and the patterns of behavior of the user in both the physical and cyber realm. Additionally, calendar events and tasks can could influence the physical location of a user or their cyber behavior within a known time window. Its transmission and storage must be done carefully to protect it from possible threats, such as eavesdropping, replay, message insertion, deletion, modification, and on-path attacks.

The data being stored and transmitted may be used in systems with real world consequences. For example, a home automation system may turn an alarm on and off. Or a coworking space may charge money to the organiser of an event that books one of their meeting rooms. Such systems must be careful to authenticate all data they receive to prevent them from being subverted, and ensure the change comes from an authorized entity.

This document just defines the data format; such considerations are primarily the concern of the API or method of storage and transmission of such files.

7.1. Expanding Recurrences

A recurrence rule may produce infinite occurrences of an event. Implementations MUST handle expansions carefully to prevent accidental or deliberate resource exhaustion.

Conversely, a recurrence rule may be specified that does not expand to anything. It is not always possible to tell this through static analysis of the rule, so implementations MUST be careful to avoid getting stuck in infinite loops, or otherwise exhausting resources while searching for the next occurrence.

Events recur in the event’s time zone. If the user is in a different time zone, daylight saving transitions may cause an event that normally occurs at, for example, 9am to suddenly shift an hour

earlier. This may be used in an attempt to cause a participant to miss an important meeting. User agents must be careful to translate date-times correctly between time zones and may wish to call out unexpected changes in the time of a recurring event.

7.2. JSON Parsing

The Security Considerations of [RFC8259] apply to the use of JSON as the data interchange format.

As for any serialization format, parsers need to thoroughly check the syntax of the supplied data. JSON uses opening and closing tags for several types and structures, and it is possible that the end of the supplied data will be reached when scanning for a matching closing tag; this is an error condition, and implementations need to stop scanning at the end of the supplied data.

JSON also uses a string encoding with some escape sequences to encode special characters within a string. Care is needed when processing these escape sequences to ensure that they are fully formed before the special processing is triggered, with special care taken when the escape sequences appear adjacent to other (non-escaped) special characters or adjacent to the end of data (as in the previous paragraph).

If parsing JSON into a non-textual structured data format, implementations may need to allocate storage to hold JSON string elements. Since JSON does not use explicit string lengths, the risk of denial of service due to resource exhaustion is small, but implementations may still wish to place limits on the size of allocations they are willing to make in any given context, to avoid untrusted data causing excessive memory allocation.

7.3. URI Values

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

Fetching remote resources carries inherent risks. Connections must only be allowed on well known ports, using allowed protocols (generally just HTTP/HTTPS on their default ports). The URL must be resolved externally and not allowed to access internal resources. Connecting to an external source reveals IP (and therefore generally location) information.

A maliciously constructed JSCalendar object may contain a very large number of URIs. In the case of published calendars with a large

number of subscribers, such objects could be widely distributed. Implementations should be careful to limit the automatic fetching of linked resources to reduce the risk of this being an amplification vector for a denial-of-service attack.

7.4. Spam

Calendar systems may receive JSCalendar files from untrusted sources, in particular as attachments to emails. This can be a vector for an attacker to inject spam into a user's calendar. This may confuse, annoy, and mislead users, or overwhelm their calendar with bogus events, preventing them from seeing legitimate ones.

Heuristic, statistical or machine-learning-based filters can be effective in filtering out spam. Authentication mechanisms such as DKIM [RFC6376] can help establish the source of messages and associate the data with existing relationships (such as an address book contact). Misclassifications are always possible, however, and providing a mechanism for users to quickly correct this is advised.

Confusable unicode characters may be used to trick a user into trusting a JSCalendar file that appears to come from a known contact but is actually from a similar-looking source controlled by an attacker.

7.5. Duplication

It is important for calendar systems to maintain the UID of an event when updating it to avoid unexpected duplication of events. Consumers of the data may not remove the previous version of the event if it has a different UID. This can lead to a confusing situation for the user, with many variations of the event and no indication of which one is correct. Care must be taken by consumers of the data to remove old events where possible to avoid an accidental denial-of-service attack due to the volume of data.

7.6. Time Zones

Events recur in a particular time zone. When this differs from the user's current time zone, it may unexpectedly cause an occurrence to shift in time for that user due to a daylight savings change in the event's time zone. A maliciously crafted event could attempt to confuse users with such an event to ensure a meeting is missed.

8. IANA Considerations

8.1. Media Type Registration

This document defines a 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: While JSCalendar is designed to avoid ambiguities as much as possible, when converting objects from other calendar formats to/from JSCalendar it is possible that differing representations for the same logical data might arise, or ambiguities in interpretation. The semantic equivalence of two JSCalendar objects may be determined differently by different applications, for example where URL values differ in case between the two objects.

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: A JSON Pointer fragment identifier may be used, as defined in [RFC6901], Section 6.

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:
calsify@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

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

Change controller: IETF

8.2. Creation of "JSCalendar Properties" Registry

The IANA will create the "JSCalendar Properties" registry to allow interoperability of extensions to JSCalendar objects.

This registry follows the Expert Review process ([RFC8126], Section 4.5). If the "intended use" field is "common", sufficient documentation is required to enable interoperability. Preliminary community review for this registry is optional but strongly encouraged.

A registration can have an intended use of "common", "reserved", or "obsolete". The IANA will list common-use registrations prominently and separately from those with other intended use values.

A "reserved" registration reserves a property name without assigning semantics to avoid name collisions with future extensions or protocol use.

An "obsolete" registration denotes a property that is no longer expected to be added by up-to-date systems. A new property has probably been defined covering the obsolete property's semantics.

The JSCalendar property registration procedure is not a formal standards process but rather an administrative procedure intended to allow community comment and sanity checking without excessive time delay. It is designed to encourage vendors to document and register new properties they add for use cases not covered by the original specification, leading to increased interoperability.

8.2.1. Preliminary Community Review

Notice of a potential new registration SHOULD be sent to the Calext mailing list <calsify@ietf.org> for review. This mailing list is appropriate to solicit community feedback on a proposed new property.

Properties registrations must be marked with their intended use: "common", "reserved" or "obsolete".

The intent of the public posting to this list is to solicit comments and feedback on the choice of the property name, the unambiguity of the specification document, and a review of any interoperability or security considerations. The submitter may submit a revised registration proposal or abandon the registration completely at any time.

8.2.2. Submit Request to IANA

Registration requests can be sent to <iana@iana.org>.

8.2.3. Designated Expert Review

The primary concern of the designated expert (DE) is preventing name collisions and encouraging the submitter to document security and privacy considerations. For a common-use registration, the DE is expected to confirm that suitable documentation, as described in Section 4.6 of [RFC8126], is available to ensure interoperability. That documentation will usually be in an RFC, but simple definitions are likely to use a web/wiki page, and if a sentence or two is deemed sufficient it could be described in the registry itself. The DE should also verify that the property name does not conflict with work that is active or already published within the IETF. A published specification is not required for reserved or obsolete registrations.

The DE will either approve or deny the registration request and publish a notice of the decision to the Calext WG mailing list or its successor, as well as inform IANA. A denial notice must be justified by an explanation, and, in the cases where it is possible, concrete suggestions on how the request can be modified so as to become acceptable should be provided.

8.2.4. Change Procedures

Once a JSCalendar property has been published by the IANA, the change controller may request a change to its definition. The same procedure that would be appropriate for the original registration request is used to process a change request.

JSCalendar property registrations may not be deleted; properties that are no longer believed appropriate for use can be declared obsolete by a change to their "intended use" field; such properties will be clearly marked in the lists published by the IANA.

Significant changes to a JSCalendar property's definition should be requested only when there are serious omissions or errors in the published specification, as such changes may cause interoperability issues. When review is required, a change request may be denied if it renders entities that were valid under the previous definition invalid under the new definition.

The owner of a JSCalendar property may pass responsibility to another person or agency by informing the IANA; this can be done without discussion or review.

8.2.5. JSCalendar Properties Registry Template

- Property Name: The name of the property. The property name MUST NOT already be registered for any of the object types listed in the "Property Context" field of this registration. Other object types MAY already have registered a different property with the same name, however the same name SHOULD only be used when the semantics are analogous.

- Property Type: The type of this property, using type signatures as specified in Section 1.3. The property type MUST be registed in the Type Registry.

- Property Context: A comma-separated list of JSCalendar object types this property is allowed on.

- Reference or Description: A brief description or RFC number and section reference where the property is specified (omitted for "reserved" property names).

- Intended Use: Common, reserved, or obsolete.

- Change Controller: Who may request a change to this entry's definition ("IETF" for IETF-stream RFCs).

8.2.6. Initial Contents for the JSCalendar Properties Registry

The following table lists the initial entries of the JSCalendar Properties registry. All properties are for common-use. All RFC section references are for this document. The change controller for all these properties is "IETF".

Property Name	Property Type	Property Context	Reference or Description
@type	String	JSEvent, JSTask, JSGroup, AbsoluteTrigger, Alert, Link, Location, NDay, OffsetTrigger, Participant, RecurrenceRule, Relation, TimeZone, TimeZoneRule, VirtualLocation	Section 4.1.1, Section 4.5.2, Section 1.4.11, Section 4.2.5, Section 4.4.5, Section 4.3.2, Section 4.1.3, Section 4.7.2, Section 4.7.2, Section 4.2.6
acknowledged	UTCDateTime	Alert	Section 4.5.2
action	String	Alert	Section 4.5.2
alerts	Id[Alert]	JSEvent, JSTask	Section 4.5.2
aliases	String[Boolean]	TimeZone	Section 4.7.2
byDay	NDay[]	RecurrenceRule	Section 4.3.2
byHour	UnsignedInt[]	RecurrenceRule	Section 4.3.2
byMinute	UnsignedInt[]	RecurrenceRule	Section 4.3.2

byMonth	String[]	Recurrence Rule	Section 4.3.2
byMonthDay	Int[]	Recurrence Rule	Section 4.3.2
bySecond	UnsignedInt[]	Recurrence Rule	Section 4.3.2
bySetPosition	Int[]	Recurrence Rule	Section 4.3.2
byWeekNo	Int[]	Recurrence Rule	Section 4.3.2
byYearDay	Int[]	Recurrence Rule	Section 4.3.2
categories	String[Boolean]	JSEvent, JSTask, JSGroup	Section 4.2.10
cid	String	Link	Section 1.4.11
color	String	JSEvent, JSTask, JSGroup	Section 4.2.11
comments	String[]	TimeZoneRule	Section 4.7.2
contentType	String	Link	Section 1.4.11
coordinates	String	Location	Section 4.2.5
count	UnsignedInt	Recurrence Rule	Section 4.3.2
created	UTCDateTime	JSEvent, JSTask, JSGroup	Section 4.1.5
day	String	NDay	Section 4.3.2

daylight	TimeZoneRule	TimeZone	Section 4.7.2
delegatedFrom	String[Boolean]	Participant	Section 4.4.5
delegatedTo	String[Boolean]	Participant	Section 4.4.5
description	String	JSEvent, JSTask, Location, Participant, VirtualLocation	Section 4.2.2, Section 4.2.5, Section 4.4.5, Section 4.2.6
descriptionContentType	String	JSEvent, JSTask	Section 4.2.3
display	String	Link	Section 1.4.11
due	LocalDateTime	JSTask	Section 5.2.1
duration	Duration	JSEvent	Section 5.1.2
email	String	Participant	Section 4.4.5
entries	(JSTask|JSEvent)[]	JSGroup	Section 5.3.1
estimatedDuration	Duration	JSTask	Section 5.2.3
excluded	Boolean	JSEvent, JSTask	Section 4.3.5
excludedRecurrenceRules	RecurrenceRule[]	JSEvent, JSTask	Section 4.3.3
expectReply	Boolean	Participant	Section 4.4.5

firstDayOfWeek	String	RecurrenceRule	Section 4.3.2
freeBusyStatus	String	JSEvent, JSTask	Section 4.4.2
frequency	String	RecurrenceRule	Section 4.3.2
href	String	Link	Section 1.4.11
interval	UnsignedInt	RecurrenceRule	Section 4.3.2
invitedBy	String	Participant	Section 4.4.5
keywords	String[Boolean]	JSEvent, JSTask, JSGroup	Section 4.2.9
kind	String	Participant	Section 4.4.5
language	String	Participant	Section 4.4.5
links	Id[Link]	JSGroup, JSEvent, JSTask, Location, Participant	Section 4.2.7, Section 4.2.5, Section 4.4.5
locale	String	JSGroup, JSEvent, JSTask	Section 4.2.8
localizations	String[PatchObject]	JSEvent, JSTask	Section 4.6.1
locationId	String	Participant	Section 4.4.5
locations	Id[Location]	JSEvent, JSTask	Section 4.2.5

locationTypes	String[Boolean]	Location	Section 4.2.5
memberOf	String[Boolean]	Participant	Section 4.4.5
method	String	JSEvent, JSTask	Section 4.1.8
name	String	Location, VirtualLocation, Participant	Section 4.2.5, Section 4.2.6, Section 4.4.5
names	String[Boolean]	TimeZoneRule	Section 4.7.2
nthOfPeriod	Int	NDay	Section 4.3.2
offset	SignedDuration	OffsetTrigger	Section 4.5.2
offsetFrom	UTCDateTime	TimeZoneRule	Section 4.7.2
offsetTo	UTCDateTime	TimeZoneRule	Section 4.7.2
participants	Id[Participant]	JSEvent, JSTask	Section 4.4.5
participationComment	String	Participant	Section 4.4.5
participationStatus	String	Participant	Section 4.4.5
percentComplete	UnsignedInt	JSTask, Participant	Section 5.2.4
priority	Int	JSEvent, JSTask	Section 4.4.1
privacy	String	JSEvent,	Section

		JSTask	4.4.3
prodId	String	JSEvent, JSTask, JSGroup	Section 4.1.4
progress	String	JSTask, Participant	Section 5.2.5
progressUpdated	UTCDateTime	JSTask, Participant	Section 5.2.6
recurrenceId	LocalDateTime	JSEvent, JSTask	Section 4.3.1
recurrenceOverrides	LocalDateTime[PatchObject]	JSEvent, JSTask, TimeZoneRule	Section 4.3.4, Section 4.7.2
recurrenceRules	RecurrenceRule[]	JSEvent, JSTask, TimeZoneRule	Section 4.3.2, Section 4.7.2
rel	String	Link	Section 1.4.11
relatedTo	String[Relation]	JSEvent, JSTask, Alert	Section 4.1.3, Section 4.5.2
relation	String[Boolean]	Relation	Section 1.4.10
relativeTo	String	OffsetTriger, Location	Section 4.5.2, Section 4.2.5
replyTo	String[String]	JSEvent, JSTask	Section 4.4.4
roles	String[Boolean]	Participant	Section 4.4.5
rscale	String	Recurrence	Section

		Rule	4.3.2
standard	TimeZoneRule	TimeZone	Section 4.7.2
start	LocalDateTime	TimeZoneRule	Section 4.7.2
scheduleAgent	String	Participant	Section 4.4.5
scheduleForceSend	Boolean	Participant	Section 4.4.5
scheduleSequence	UnsignedInt	Participant	Section 4.4.5
scheduleStatus	String[]	Participant	Section 4.4.5
scheduleUpdated	UTCDateTime	Participant	Section 4.4.5
sendTo	String[String]	Participant	Section 4.4.5
sequence	UnsignedInt	JSEvent, JSTask	Section 4.1.7
showWithoutTime	Boolean	JSEvent, JSTask	Section 4.2.4
size	UnsignedInt	Link	Section 1.4.11
skip	String	RecurrenceRule	Section 4.3.2
source	String	JSGroup	Section 5.3.2
start	LocalDateTime	JSEvent, JSTask	Section 5.1.1, Section 5.2.2
status	String	JSEvent	Section 5.1.3

timeZone	TimeZoneId|null	JSEvent, JSTask, Location	Section 4.7.1, Section 4.2.5
timeZones	TimeZoneId[TimeZone]	JSEvent, JSTask	Section 4.7.2
title	String	JSEvent, JSTask, JSGroup, Link	Section 4.2.1
trigger	OffsetTrigger|AbsoluteTrigger|UnknownTrigger	Alert	Section 4.5.2
tzId	String	TimeZone	Section 4.7.2
uid	String	JSEvent, JSTask, JSGroup	Section 4.1.2
until	LocalDateTime	RecurrenceRule	Section 4.3.2
updated	UTCDateTime	JSEvent, JSTask, JSGroup	Section 4.1.6
uri	String	VirtualLocation	Section 4.2.6
url	String	TimeZone	Section 4.7.2
useDefaultAlerts	Boolean	JSEvent, JSTask	Section 4.5.1
validUntil	UTCDateTime	TimeZone	Section 4.7.2
virtualLocations	Id[VirtualLocation]	JSEvent, JSTask	Section 4.2.6
when	UTCDateTime	AbsoluteTrigger	Section 4.5.2

+--------------+---------------------------+-----------+---------+

Table 1

8.3. Creation of "JSCalendar Types" Registry

The IANA will create the "JSCalendar Types" registry to avoid name collisions and provide a complete reference for all data types used for JSCalendar property values. The registration process is the same as for the JSCalendar Properties registry, as defined in Section 8.2.

8.3.1. JSCalendar Types Registry Template

- Type Name: The name of the type.
- Reference or Description: A brief description or RFC number and section reference where the Type is specified (may be omitted for "reserved" type names).
- Intended Use: Common, reserved, or obsolete.
- Change Controller: Who may request a change to this entry's definition ("IETF" for IETF-stream RFCs).

8.3.2. Initial Contents for the JSCalendar Types Registry

The following table lists the initial entries of the JSCalendar Types registry. All properties are for common-use. All RFC section references are for this document. The change controller for all these properties is "IETF".

Type Name	Reference or Description
Alert	Section 4.5.2
Boolean	Section 1.3
Duration	Section 1.4.6
Id	Section 1.4.1
Int	Section 1.4.2
LocalDateTime	Section 1.4.5
Link	Section 1.4.11

Location	Section 4.2.5
NDay	Section 4.3.2
Number	Section 1.3
Participant	Section 4.4.5
PatchObject	Section 1.4.9
RecurrenceRule	Section 4.3.2
Relation	Section 1.4.10
SignedDuration	Section 1.4.7
String	Section 1.3
TimeZone	Section 4.7.2
TimeZoneId	Section 1.4.8
TimeZoneRule	Section 4.7.2
UnsignedInt	Section 1.4.3
UTCDateTime	Section 1.4.4
VirtualLocation	Section 4.2.6

Table 2

8.4. Creation of "JSCalendar Enum Values" Registry

The IANA will create the "JSCalendar Enum Values" registry to allow interoperable extension of semantics for properties with enumerable values. Each such property will have a subregistry of allowed values. The registration process for a new enum value or adding a new enumerable property is the same as for the JSCalendar Properties registry, as defined in Section 8.2.

8.4.1. JSCalendar Enum Property Template

This template is for adding a subregistry for a new enumerable property to the JSCalendar Enum registry.

- Property Name: the name(s) of the property or properties where these values may be used. This MUST be registered in the JSCalendar Properties registry.

- Context: the list of allowed object types where the property or properties may appear, as registered in the JSCalendar Properties registry. This disambiguates where there may be two distinct properties with the same name in different contexts.

- Change Controller: ("IETF" for properties defined in IETF-stream RFCs).

- Initial Contents: The initial list of defined values for this enum, using the template defined in Section 8.4.2. A subregistry will be created with these values for this property name/context tuple.

8.4.2. JSCalendar Enum Value Template

This template is for adding a new enum value to a subregistry in the JSCalendar Enum registry.

- Enum Value: The verbatim value of the enum.

- Reference or Description: A brief description or RFC number and section reference for the semantics of this value.

8.4.3. Initial Contents for the JSCalendar Enum Values registry

For each subregistry created in this section, all RFC section references are for this document.

Property Name: action

Context: Alert

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
display	Section 4.5.2
email	Section 4.5.2

Table 3

Property Name: display

Context: Link

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
badge	Section 1.4.11
graphic	Section 1.4.11
fullsize	Section 1.4.11
thumbnail	Section 1.4.11

Table 4

Property Name: freeBusyStatus

Context: JSEvent, JSTask

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
free	Section 4.4.2
busy	Section 4.4.2

Table 5

Property Name: kind

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
individual	Section 4.4.5
group	Section 4.4.5
resource	Section 4.4.5
location	Section 4.4.5

Table 6

Property Name: participationStatus

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
needs-action	Section 4.4.5
accepted	Section 4.4.5
declined	Section 4.4.5
tenative	Section 4.4.5
delegated	Section 4.4.5

Table 7

Property Name: privacy

Context: JSEvent, JSTask

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
public	Section 4.4.3
private	Section 4.4.3
secret	Section 4.4.3

Table 8

Property Name: progress

Context: JSTask, Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
needs-action	Section 5.2.5
in-process	Section 5.2.5
completed	Section 5.2.5
failed	Section 5.2.5
cancelled	Section 5.2.5

Table 9

Property Name: relation

Context: Relation

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
first	Section 1.4.10
next	Section 1.4.10
child	Section 1.4.10
parent	Section 1.4.10

Table 10

Property Name: relativeTo

Context: OffsetTrigger, Location

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
start	Section 4.5.2
end	Section 4.5.2

Table 11

Property Name: roles

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
owner	Section 4.4.5
attendee	Section 4.4.5
optional	Section 4.4.5
informational	Section 4.4.5
chair	Section 4.4.5
contact	Section 4.4.5

Table 12

Property Name: scheduleAgent

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
server	Section 4.4.5
client	Section 4.4.5
none	Section 4.4.5

Table 13

Property Name: status

Context: JSEvent

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
confirmed	Section 5.1.3
cancelled	Section 5.1.3
tentative	Section 5.1.3

Table 14

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

[CLDR] "Unicode Common Locale Data Repository", <http://cldr.unicode.org/>.

[COLORS] "CSS Color Module", <https://www.w3.org/TR/css-color-3/>.

[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>.

[RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, DOI 10.17487/RFC2397, August 1998, <https://www.rfc-editor.org/info/rfc2397>.

[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>.

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

[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>.

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

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <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>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[TZDB] "IANA Time Zone Database", <https://www.iana.org/time-zones>.

10.2. Informative References

[LINKRELS] "IANA Link Relation Types", <https://www.iana.org/assignments/link-relations/link-relations.xhtml>.

[LOCATIONTYPES] "IANA Location Types Registry", <https://www.iana.org/assignments/location-type-registry/location-type-registry.xhtml>.

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

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[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>.

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: Standards Track
Expires: August 25, 2021

N. Jenkins
R. Stepanek
FastMail
M. Douglass
BCS

February 21, 2021

JSCalendar: Converting from and to iCalendar
draft-ietf-calext-jscalendar-icalendar-04

Abstract

This document provides the required methods 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 August 25, 2021.

Copyright Notice

Copyright (c) 2021 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

1.1. Note (to be deleted later)

This is still very much a work in progress. There are implementations of the mapping but there may be changes over the coming weeks.

1.2. Motivation

The JSCalendar [draft-ietf-calext-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, there is a requirement to preserve custom iCalendar properties that have no equivalent in JSCalendar when converting between these formats.

To support these use cases, this document provides the required approach when converting JSCalendar data from and to iCalendar.

1.3. Scope and caveats

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

- 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. These values may have to be extracted from a full copy of the iCalendar format provided as a property in the JSCalendar data.

- iCalendar may contain arbitrary, non-standardised data with custom properties/attributes. These will be translated using the same approach as jCal.

- 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.

- 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 defines a canonical translation between iCalendar and JSCalendar, and implementations MUST follow the approaches specified here when iCalendar data is represented in JSCalendar and vice-versa.

This document defines mappings for the following specifications.

- Internet Calendaring and Scheduling Core Object Specification (iCalendar)
- iCalendar Transport-Independent Interoperability Protocol (iTIP)
- New Properties for iCalendar
- Event Publishing Extensions to iCalendar
- Support for iCalendar Relationships
- VALARM Extensions for iCalendar

Therefore all of these specifications MUST be implemented to follow this specification.

1.4. 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. FRACTIONAL parameter

Parameter name: FRACTIONAL

Purpose: This parameter is used to contain a value with fractional seconds for time values and durations. FRACTIONAL 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. It MUST be a valid iCalendar DATE-TIME or

DURATION value with the addition of fractional seconds. The value MUST NOT be negative for durations but MAY be negative for alarm triggers. iCalendar implementations SHOULD ignore this parameter in date time arithmetic. Implementations MUST ignore presence of the FRACTIONAL parameter on RECURRENCE-ID properties when determining recurrence overrides.

Applications receiving a property with a FRACTIONAL parameter MUST ensure its value is consistent with the value of the property. The property value must match:
- a positive FRACTIONAL value rounded up to the next non-fractional second or
- a negative FRACTIONAL value rounded down the next non-fractional second

If the values do not match the the application MUST assume that the property value has been updated by an application that is unaware of the FRACTIONAL parameter. The parameter should be ignored in this case.

Format Definition:

This parameter is defined by the following notation:

```
fractional-param = DATE-TIME or DURATION
```

Example:

```
DTSTART;FRACTIONAL=20190605T133015.03:20190605T133015
```

3. iCalendar pre-processing

iCalendar uses a line-folding mechanism to limit lines of data to a maximum line length (typically 75 octets) to ensure the maximum likelihood of preserving data integrity as it is transported via various means (e.g., email) -- see Section 3.1 of [RFC5545].

iCalendar data uses an "escape" character sequence for text values and property parameter values. See Sections 3.1 and 3.3 of [RFC5545] as well as [RFC6868].

There is a subtle difference in the number representations between JSON and iCalendar. While in iCalendar, a number may have leading zeros, as well as a leading plus sign; this is not the case in JSON. Numbers should be represented in whatever way needed for the underlying format.

When converting from iCalendar to JSCalendar: First, iCalendar lines MUST be unfolded. Afterwards, any iCalendar escaping MUST be unescaped. Finally, JSON escaping, as described in Section 7 of [RFC7159], MUST be applied. The reverse order applies when converting from JSCalendar to iCalendar, which is further described in Section ?.

iCalendar uses a base64 encoding for binary data. However, it does not restrict the encoding from being applied to non-binary value types. So, the following rules are applied when processing a property with the "ENCODING" property parameter set to "BASE64":

- If the property value type is "BINARY", the base64 encoding MUST be preserved.

- If the value type is not "BINARY", the "ENCODING" property parameter MUST be removed, and the value MUST be base64 decoded.

When base64 encoding is used, it MUST conform to Section 4 of [RFC4648], which is the base64 method used in [RFC5545].

One key difference in the formatting of values used in iCalendar and JSCalendar is that, in JSCalendar, the specification uses date/time values aligned with the extended format of [ISO.8601.2004], which is more commonly used in Internet applications that make use of the JSON format. The sections of this document describing the various date and time formats contain more information on the use of the complete representation, reduced accuracy, or truncated representation.

4. Translating iCalendar components to JSCalendar

This section is an alphabetic list of [RFC5545] components and how they are mapped to JSCalendar.

At present VFREEBUSY and VJOURNAL are not mapped in jscalendar.

4.1. VALARM

An [RFC5545] VALARM component is mapped to a member of a JSCalendar "alerts" object with a type of "Alert" and a small id.

```
BEGIN: VEVENT
...
BEGIN: VALARM
...
END: VALARM
BEGIN: VALARM
...
END: VALARM
END: VEVENT

maps to

{
  "@type": "jsevent",
  ...
  "alerts": {
    "1": {
      "@type": "Alert",
      ...
    },
    "2": {
      "@type": "Alert",
      ...
    }
  }
}
```

The [RFC5545] VALARM has a number of problems which are not carried over into JSCalendar. Clients tend to choose how, and in some cases when to notify the user.

For example, if the user has a smart-watch they may get tapped on the wrist. The method of notification may depend on which device is being used and the context, for example a meeting or driving.

Also, many clients are taking into consideration the travel time and notifying the user earlier if it seems necessary.

Specifying that a client should send emails to all attendees is both annoying and dangerous. Attendees have their own preferences for how and when they should be notified.

Accordingly, the specification only allows for "display" and "email" actions and - other than specifying when - does not allow much else. Clients and/or servers will generally use the associated event or task title as identification. User preferences generally indicate what actions they prefer.

An [RFC5545] ACTION property can take the defined values "AUDIO" / "DISPLAY" / "EMAIL" whereas the JSCalendar "action" property only supports "display" and "email".

An "AUDIO" alarm SHOULD be mapped to a "display" alert. Any attachment MUST be ignored.

The [RFC5545] example VALARMs will be mapped as follows, assuming they are all in the same event:

```
BEGIN:VEVENT
...
BEGIN:VALARM
TRIGGER;VALUE=DATE-TIME:19970317T133000Z
REPEAT:4
DURATION:PT15M
ACTION:AUDIO
ATTACH;FMTTYPE=audio/basic:ftp://example.com/pub/
sounds/bell-01.aud
END:VALARM
BEGIN:VALARM
TRIGGER:-PT30M
REPEAT:2
DURATION:PT15M
ACTION:DISPLAY
DESCRIPTION:Breakfast meeting with executive\n
 team at 8:30 AM EST.
END:VALARM
BEGIN:VALARM
TRIGGER;RELATED=END:-P2D
ACTION:EMAIL
ATTENDEE:mailto:john_doe@example.com
SUMMARY:*** REMINDER: SEND AGENDA FOR WEEKLY STAFF MEETING ***
DESCRIPTION:A draft agenda needs to be sent out to the attendees
to the weekly managers meeting (MGR-LIST). Attached is a
pointer the document template for the agenda file.
ATTACH;FMTTYPE=application/msword:http://example.com/
templates/agenda.doc
END:VALARM
END:VEVENT
```

maps to

```
{
  "@type": "jsevent",
  ...
  "alerts": {
    "1": {
```

```
          "@type": "Alert",
          "action": "display",
          "trigger": {
            "@type": "AbsoluteTrigger",
            "when": "19970317T133000Z"
          }
        },
        "2": {
          "@type": "Alert",
          "action": "display",
          "trigger": {
            "@type": "OffsetTrigger",
            "offset": "-PT30M"
          }
        }
        "3": {
          "@type": "Alert",
          "action": "email",
          "trigger": {
            "@type": "OffsetTrigger",
            "offset": "-P2D",
            "relativeTo": "end"
          }
        }
      }
    }
```

Note that the ATTACH, ATTENDEE, DESCRIPTION, DURATION, REPEAT and SUMMARY properties have been dropped.

4.2. VCALENDAR

A [RFC5545] VCALENDAR component may be mapped to a JSCalendar object with a type of "jsgroup".

```
  BEGIN: VCALENDAR
  PRODID:-//ABC Corporation//NONSGML My Product//EN
  ...
  END: VCALENDAR

  maps to

  {
    "@type": "jsgroup",
    "prodid": "-//ABC Corporation//NONSGML My Product//EN",
    ...
  }
```

Note that a single JSEvent or JSTask MAY be converted without a surrounding JSGroup if the VCALENDAR component only contains PRODID and CALSCALE properties. In this case the prddid can go in the JSEvent or JSTask. The CALSCALE property is dropped - there is no equivalence in JSCalendar.

4.3. VEVENT

A [RFC5545] VEVENT component is mapped to a JSCalendar object with a type of "jsevent".

```
BEGIN: VEVENT
...
END: VEVENT

maps to

{
  "@type": "jsevent",
  ...
}
```

4.4. VLOCATION

A [draft-ietf-calext-eventpub-extensions] VLOCATION component is mapped to a JSCalendar object with a type of "Location". Any properties within the VLOCATION must be mapped as described below.

```
BEGIN: VEVENT
...
BEGIN: VLOCATION
...
END: VLOCATION
END: VEVENT

maps to

{
  "@type": "jsevent",
  ...
   "locations": {
     "1": {
       "@type": "Location",
       ...
     }
   }
}
```

4.5. VTIMEZONE, STANDARD, DAYLIGHT

A [RFC5545] VTIMEZONE component is mapped to a member of a JSCalendar "timezones" object with a type of "TimeZone" and an id which follows the restrictions specified.

The STANDARD and DAYLIGHT components map to JSCalendar TimeZoneRule objects as members of the

Note that

- There is no current approach for defining standalone sets of timezones.
- Timezones defined in the IANA timezone database SHOULD NOT be redefined in the object. Only custom timezones will be defined.

```
BEGIN: VTIMEZONE
TZID: Example/Somewhere
...
END: VTIMEZONE
BEGIN: VTIMEZONE
TZID: Example/Somewhere-else
...
END: VTIMEZONE
BEGIN: VEVENT
...
END: VEVENT
```

maps to

```
{
  "@type": "jsevent",
  ...
  "timezones": {
    "/Example/Somewhere: {
      "@type": "TimeZone",
      ...
    },
    "/Example/Somewhere-else": {
      "@type": "TimeZone",
      ...
    }
  }
}
```

4.6. VTODO

A [RFC5545] VTODO component is mapped to a JSCalendar object with a type of "jstask".

```
BEGIN: VTODO
...
END: VTODO

maps to

{
  "@type": "jstask",
  ...
}
```

5. Translating iCalendar properties to JSCalendar

This section is an alphabetic list of [RFC5545] and [RFC7986] properties and how they are mapped to JSCalendar.

5.1. ATTACH

A [RFC5545] ATTACH allows for two types of attachment:

- A uri value
- A binary value

Both map to a JSCalendar "link" object with a "rel" of "enclosure" and the "href" set to the value of the property.

If the FMTTYPE parameter is set then add a JSCalendar "contentType" property to the link object.

For a binary value use a base64 data uri.

For an example of a recurring event with ATTACH see Section 7.1

5.2. ATTENDEE

An [RFC5545] ATTENDEE maps to the JSCalendar "participant" property with a JSCalendar "role" of "attendee". The value for role should always be set.

In the simplest case a JSCalendar "participant" property will be created and added to the JSCalendar "participants" property.

The value of the ATTENDEE property is used to add an "imip" method to the JSCalendar "sendTo" property. The value of the entry will be the ATTENDEE property value.

For example:

```
...
ATTENDEE:mailto:user01@example.org
...
```

maps to

```
{
...
    "participants": {
      "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
        "@type": "Participant",
        "sendTo": {
          "imip": "user01@example.org"
        },
...
}
```

The attendee parameters are mapped to JSCalendar "participant" properties as follows:

CN: The value of the CN parameter is used to set the JSCalendar "name" property.

CUTYPE: This maps on to the JSCalendar "kind" property as follows:

INDIVIDUAL "individual"

GROUP "group"

RESOURCE "resource"

ROOM "location"

UNKNOWN No value

Any other value should be converted to lower case and assigned to the JSCalendar "kind" property.

DELEGATED-FROM: Split the value at any commas and add each resulting element to the JSCalendar "delegatedFrom" property

DELEGATED-TO: Split the value at any commas and add each resulting element to the JSCalendar "delegatedFrom" property

DIR: If non-null look in the participant "links" property for a JSCalendar "link" property with an href with the same value as the DIR parameter. You may need to search the current override and the master.

If none is found create a new one with the JSCalendar "href" property set to the value of the DIR parameter and the JSCalendar "rel" property set to "alternate"

LANG: set the JSCalendar "language" property to the value of the LANG parameter.

MEMBER: If this is set there should be a corresponding ATTENDEE object with a value equal to the value of the member parameter. If not it is appropriate to skip this parameter.

If there is a corresponding ATTENDEE then there should be a corresponding JSCalendar "participant" property. This suggests that CUTYPE=GROUP ATTENDEE properties should be processed ahead of the others.

Locate the JSCalendar "participant" property for the group. This may be in the current override or in the master. Add the id to the current participants JSCalendar "memberOf" property.

PARTSTAT: If the PARTSTAT parameter is set and is not "NEEDS-ACTION" then set the JSCalendar "participationStatus" property to the lower-cased value of the PARTSTAT.

ROLE: This is mapped to the JSCalendar "roles" property as follows:

CHAIR "attendee" and "chair"

REQ-PARTICIPANT "attendee"

OPT-PARTICIPANT "attendee" and "optional"

NON-PARTICIPANT "informational"

Any other value should be converted to lower case and added to the JSCalendar "roles" property.

RSVP: If the value of the RSVP parameter is TRUE set the JSCalendar "expectReply" property to "true" otherwise omit it.

SCHEDULE-AGENT: If the value is "CLIENT" (ignoring case) set the JSCalendar "scheduleAgent" property to "client" otherwise omit it.

SCHEDULE-FORCE-SEND: Set the JSCalendar "scheduleForceSend" property to the lower-cased value of the [RFC6638] SCHEDULE-FORCE-SEND parameter.

SCHEDULE-STATUS: Split the value at any commas and add each resulting element to the JSCalendar "scheduleStatus" property.

SENT-BY: The value of the SENT-BY parameter is used to set the JSCalendar "invitedBy" property.

5.3. CALSCALE

A [RFC5545] CALSCALE has no equivalence in JSCalendar. It is ignored.

5.4. CATEGORIES

These map on to the JSCalendar "keywords" property with each category being the key to an entry.

```
...
CATEGORIES:APPOINTMENT,EDUCATION
CATEGORIES:MEETING
...
```

maps to

```
...
"keywords": {
     "APPOINTMENT": true,
     "EDUCATION": true,
     "MEETING": true
   },
...
```

5.5. CLASS

Maps to the "privacy" property. The iCalendar property value maps to the JSCalendar value as follows:

CONFIDENTIAL "secret"

PRIVATE "private"

PUBLIC "public"

iana-token and x-name verbatim copy

5.6. COLOR

Maps to the "color" property. Copy the verbatim value.

5.7. COMMENT

There is no direct mapping for this property which may appear multiple times in [RFC5545].

For a scheduling reply it is presumably a message by the participant so the value or values should be used to set the JSCalendar "participantComment" property.

5.8. COMPLETED

Set the JSCalendar "progress" property to "completed" and the "progressUpdated" property to the reformatted date/time.

```
...
COMPLETED: "20101010T101010Z"
...
```

maps to

```
...
"progressUpdated": "2010-10-10T10:10:10Z",
"progress": "completed",
...
```

5.9. CONCEPT

This [draft-ietf-calext-ical-relations] property may appear multiple times in components.

Each instance of the property is mapped on to a member of the JSCalendar "categories" property.

```
...
CONCEPT:http://example.com/event-types/arts/music
CONCEPT:http://example.com/performance-types/arts/live
...

maps to

...
"categories": {
  "http://example.com/event-types/arts/music": true,
  "http://example.com/performance-types/arts/live": true
}
...
```

5.10. CONFERENCE

Maps to a "VirtualLocation" object. The property value maps to the "uri" property of the virtual location.

Mapping parameters:

FEATURE: Maps to the "features" property of the virtual location.

LABEL: Maps to the "name" property of the virtual location.

LANGUAGE: No mapping.

5.11. CONTACT

The CONTACT property is mapped on to a participant object with a "roles" property of "contact" and an "order" property of 1 (one). This defines the participant as a primary contact.

Mapping parameters:

ALTREP Use the same process as for the ATTENDEE DIR parameter: create a link property with the "rel" property set to "alternate" and the "href" property set to the value of the ALTREP parameter. Then add the link to the participants "links" property.

LANG Set the participants "language" property.

For an example see Section 7.2

5.12. CREATED

The CREATED property is mapped on to a "created" property with a json formatted form of the date. Example:

```
BEGIN:VEVENT
...
CREATED:19960329T133000Z
...
END:VEVENT
```

maps to

```
{
  "@type": "jsevent",
  ...
  "created": "1996-03-29T13:30"00Z",
  ...
}
```

5.13. DESCRIPTION

Copy the value, preprocessed according to Section 3 into the "description" property.

Mapping parameters:

ALTREP No mapping.

LANG Use the "locale" property.

Example:

```
BEGIN:VEVENT
...
DESCRIPTION:We are having a meeting all this week at 12 pm fo
 r one hour\, with an additional meeting on the first day 2 h
 ours long.\nPlease bring your own lunch for the 12 pm meetin
 gs.
...
END:VEVENT
```

maps to

```
{
  "@type": "jsevent",
  ...
  "description": // Note: comments and string concatenation are not
          // allowed per the JSON specification and is used here
          // to avoid long lines.
      "We are having a meeting all this week at 12 pm for one " +
      "hour, with an additional meeting on the first day 2 " +
      "hours long.\nPlease bring your own lunch for the 12 pm " +
      "meetings.",
  ...
}
```

5.14. DTEND, DTSTART, DUE, DURATION

If the DTSTART is a DATE only property then add the JSCalendar showWithoutTime property with the value set to "true". The JSCalendar "start" property is set with zero time values.

If the DTSTART has a TZID parameter then set the JSCalendar "timeZone" property to the value of TZID.

If the DTSTART has a UTC value then set the JSCalendar "timeZone" property to the value "Etc/UTC". The JSCalendar "start" property is set without any UTC indicator.

JSCalendar has no equivalent to DTEND. If the component has a DTEND then calculate a value for "DURATION" from that property and DTSTART and proceed as below.

If the DTEND has a TZID parameter with a value that differs from the DTSTART TZID parameter then a "location" object should be created with a "relativeTo" property set to "end" and a "timezone" property set to the value of the "TZID" parameter.

Note that a task is not required to have a DTSTART so the JSCalendar "timezone" property needs to be set from the DUE property.

Convert a DURATION property to the JSCalendar duration.

Example - DTSTART and DTEND in same timezone:

```
BEGIN:VEVENT
...
DTSTART;TZID=America/New_York:20170315T150000
DTEND;TZID=America/New_York:20170315T160000

...
END:VEVENT
```

maps to

```
{
  "@type": "jsevent",
  ...
  "start": "2017-03-15T15:00:00",
  "timeZone": "America/New_York",
  "duration": "PT1H"
  ...
}
```

Example - DTSTART and DTEND in different timezone:

```
BEGIN:VEVENT
...
DTSTART;TZID=America/New_York:20170315T150000
DTEND;TZID=America/LosAngeles:20170315T190000

...
END:VEVENT

maps to

{
  "@type": "jsevent",
  ...
  "start": "2017-03-15T15:00:00",
  "timeZone": "America/New_York",
  "duration": "PT7H"
  ...
  "locations": {
     "1": {
            "@type": "location",
            "relatedTo": "end",
            "timeZone": "America/Los_Angeles"
          }
     }
}
```

Example - 3 day event:

```
BEGIN:VEVENT
...
DTSTART;VALUE=DATE:20210315
DTEND;VALUE=DATE:20210318

...
END:VEVENT

maps to

{
  "@type": "jsevent",
  ...
  "start": "2017-03-15T00:00:00",
  "duration": "P3D",
  "showWithoutTime": true,
  ...
}
```

5.15. EXDATE

Create a patch object with the recurrence id set from the EXDATE value. Add a single JSCalendar "excluded" property with the value set to true. There MUST NOT be any other properties set - other than "@type".

5.16. EXRULE

Maps to the "excludedRecurrenceRules" property. Also see Section 5.31.

5.17. DTSTAMP and LAST-MODIFIED

The mapping depends on whether or not the component is a scheduling entity.

Not a scheduling entity: The [RFC5545] DTSTAMP and LAST-MODIFIED properties have essentially the same meaning. If both are present use the value of the latest for the "updated" property. Otherwise set from whichever is present.

Is a scheduling entity: DTSTAMP should be used to set the "ScheduleUpdated" property in the "participant" object for the attendee.

If present LAST-MODIFIED should be used to set the "updated" property - otherwise set it from the DTSTAMP.

5.18. GEO

Maps to a Location object, with only the "coordinates" property set. Note that the JSCalendar coordinates property value MUST be a valid "geo" URI, so replace the ";" character in the iCalendar value with "," and prepend the resulting string with "geo:".

5.19. IMAGE

Maps to a Link object with the iCalendar property value mapped to the location "href" property, and the "rel" property set to "icon".

For a binary value use a base64 data uri in the "href" property.

Mapping parameters:

ALTREP No mapping.

FMTTYPE Maps to the "contentType" property of the Link object.

DISPLAY Maps to the "display" property of the Link object. The property values "BADGE", "GRAPHIC", "FULLSIZE" and "THUMBNAIL" map to their lower-case equivalent in JSCalendar.

5.20. LOCATION

If any [draft-ietf-calext-eventpub-extensions] "VLOCATION" components are present, then the [RFC5545]"LOCATION" property should be ignored.

To map the property create a "locations" property with a single "location" and set the "description" property to the value of the [RFC5545]"LOCATION" property.

Mapping parameters:

ALTREP Maps to a Link object in the Location "links" property, with the "href" property set to the parameter value.

5.21. METHOD

Maps to the "method" property of the JSCalendar object. The JSCalendar property value is the lowercase equivalent of the iCalendar property value.

5.22. ORGANIZER

Maps to the "replyTo" property of the JSCalendar object. An iCalendar property value in the "mailto:" URI scheme, maps to the "imip" method, any other value maps to the "other" method.

If the iCalendar component also contains an ATTENDEE with the same calendar user address then map that ATTENDEE as defined in Section 5.2 and add the "owner" role to the Participant "roles" property. Otherwise, use the ORGANIZER property to map to a Participant object. The "roles" property of the Participant MUST only contain the "owner" role and the "expectReply" property value MUST be "false". Any iCalendar parameters map as defined for ATTENDEE.

TBD: SENT-BY parameter. Example.

5.23. PERCENT-COMPLETE

For all methods other than REPLY (or no method), the PERCENT-COMPLETE applies to the VTODO as a whole. In this case it the value is used to set the JSCalendar "percentComplete" property in the task object.

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
METHOD:PUBLISH
BEGIN:VTODO
...
PERCENT-COMPLETE:39
END:VTODO
END: VCALENDAR
```

maps to

```
{
  "@type": "jstask",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
  "percentComplete": 39
}
```

PERCENT-COMPLETE in a REPLY is used to indicate the level of completeness of the ATTENDEE. There should only be a single ATTENDEE in the VTODO object.

As ever recurrences complicate matters. For a non-recurring event or an override that contains the single participant, set the JSCalendar "percentComplete" property in the JSCalendar "participant" object representing the attendee.

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
METHOD:REPLY
BEGIN:VTODO
...
ATTENDEE:mailto:douglm@example.org
PERCENT-COMPLETE:39
END:VTODO
END: VCALENDAR
```

maps to

```
{
  "@type": "jstask",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
  "participants": {
    "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
      "@type": "Participant",
      "sendTo": {
        "imip": "mailto:douglm@example.org"
      },
      "percentComplete": 39,
      "roles": {
        "attendee": true
      }
    },
    ...
}
```

In the case of an override with the participant appearing in the master then add a patch to the override.

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
METHOD:REPLY
BEGIN:VTODO
...
ATTENDEE:mailto:douglm@example.org
END:VTODO
BEGIN:VTODO
...
RECURRENCE-ID:20200523T120000
...
ATTENDEE:mailto:douglm@example.org
PERCENT-COMPLETE:39
END:VTODO
END: VCALENDAR
```

maps to

```
{
  "@type": "jstask",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
  "participants": {
    "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
      "@type": "Participant",
      "sendTo": {
        "imip": "mailto:douglm@example.org"
      },
      "roles": {
        "attendee": true
      }
    },
    "recurrenceOverrides": {
      "2020-05-23T12:00:00": {
        "participants/be4...6ce3/percentComplete": 39
      },
      ...
  }
}
```

5.24. PRIORITY

Simply copy value into the JSCalendar "priority" property.

5.25. PRODID

For a vcalendar JSGroup object with multiple JSEvent and/or JSTask object the [RFC5545] VCALENDAR PRODID is mapped to a JSCalendar "prodid" property in the group.

When mapping to a single JSEvent and/or JSTask object the [RFC5545] VCALENDAR PRODID is mapped to a JSCalendar "prodid" property in the group

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
BEGIN:VEVENT
...
END:VEVENT
END: VCALENDAR
```

maps to

```
{
  "@type": "jsevent",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
}
```

5.26. RECURRENCE-ID

Refer to Section 6 for information on mapping recurrence ids.

5.27. RELATED-TO

This is mapped to the JSCalendar "relatedTo" property which is a map of relations with the target UID as the keys. The iCalendar relation is by default a PARENT relationship. There is no default for JSCalendar so the relationship must be explicitly specified.

The RELTYPE parameter values map to their lowercase equivalents in the "relation" property.

Also note that the iCalendar relationship types are not identical. CHILD and PARENT map to JSCalendar "child" and "parent" but the best match for iCalendar SIBLING is "next"

```
...
RELATED-TO:jsmith.part7.19960817T083000.xyzMail@example.com
RELATED-TO;RELTYPE=SIBLING:
 19960401-080045-4000F192713-0052@example.com
...
```

maps to

```
"relatedTo" : {
  "jsmith.part7.19960817T083000.xyzMail@example.com" : {
    "@type" : "Relation",
    "relation" : {
      "parent" : true
    }
  },
  "19960401-080045-4000F192713-0052@example.com" : {
    "@type" : "Relation",
    "relation" : {
      "next" : true
    }
  },
},
{
  "@type": "jsevent",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
}
```

5.28. REQUEST-STATUS

Copy the value into the JSCalendar "requestStatus" property.

5.29. RESOURCES

The RESOURCES property value is a comma-separated list of resources. First split this into the separate resource names and then each resource is mapped on a participant object with a "kind" property of "resource" and the "name" property set to the resource name.

Mapping parameters:

ALTREP Use the same process as for the ATTENDEE DIR parameter: create a link property with the "rel" property set to "alternate" and the "href" property set to the value of the ALTREP parameter. Then add the link to the participants "links" property.

LANG Set the participants "language" property.

For an example see Section 7.3

5.30. RDATE

If the RDATE has a RANGE=THISANDFUTURE parameter then the recurrence MUST be split at this RDATE.

Truncate the original object before this RDATE, create a new master representing the object and link them by setting the jscalendar "relatedTo" property in both.

Otherwise create a patch object with the recurrence id set from the RDATE value. If the instance has overrides the differences will also be set in the object.

5.31. RRULE

Each RRULE is converted to an object in the JSCalendar "recurrenceRules" property. Each entry has the type "RecurrenceRule".

```
  ...
  RRULE:...
  ...

  maps to

  ...
  "recurrenceRules" : [{
      "@type" : "RecurrenceRule",
      ...
    }],
    ...
```

The recurrence rule object has one property for each element of the recurrence rule. The iCalendar rule has to be parsed out and the individual jscalendar property values set. Most take the same type but there are exceptions.

FREQ (mandatory) Copy into the jscalendar "frequency" property converted to lowercase.

INTERVAL If present and not 1 copy into the jscalendar "interval" property.

RSCALE If present copy into the jscalendar "rscale" property converted to lowercase.

SKIP If present copy into the jscalendar "skip" property converted to lowercase.

WKST If present copy into the jscalendar "firstDayOfWeek" property converted to lowercase.

BYDAY If present each element becomes an entry in the jsCalendar "byDay" propety. This is an array of NDay objects which may have 2 properties:

day The two character weekday abbreviation.

nthOfPeriod If the weekday abbreviation is preceded by a signed integer value set the jscalendar "nthOfPeriod" property.

```
...
RRULE:...,BYDAY=-1MO
...
```

maps to

```
...
"recurrenceRules" : [{
    "@type" : "RecurrenceRule",
    ...
    "byday": [{
        "day": "mo",
        "nthOfPeriod": -1
      }]
    ...
 }],
 ...
```

BYMONTHDAY If present each element will be an element in the jscalendar "byMonthDay" property.

BYMONTH If present each element will be an element in the jscalendar "byMonth" property.

Note that the iCalendar values are numeric but the JSCalendar values are strings. This is because of the possible "L" suffix for leap months.

BYYEARDAY If present each element will be an element in the jscalendar "byYearDay" property.

BYWEEKNO If present each element will be an element in the jscalendar "byWeekNo" property.

BYHOUR If present each element will be an element in the jscalendar "byHour" property.

BYMINUTE If present each element will be an element in the jscalendar "byMinute" property.

BYSECOND If present each element will be an element in the jscalendar "bySecond" property.

BYSETPOS If present each element will be an element in the jscalendar "bySetPosition" property.

COUNT If present set in the jscalendar "count" property.

UNTIL If present set the jscalendar "until" property with the appropriately reformatted value. If the is no time part append a 0 time and reformat as a jscalendar local date/time.

Some examples:

```
...
RRULE:FREQ=DAILY;COUNT=10
...
```

maps to

```
...
"recurrenceRules" : [{
    "@type" : "RecurrenceRule",
    "frequency": "daily",
    "count": 10
  }],
  ...
```

```
...
RRULE:FREQ=YEARLY;UNTIL=20220512T140000Z;
 BYMONTH=1;BYDAY=SU,MO,TU,WE,TH,FR,SA
...
```

maps to

```
...
"recurrenceRules" : [{
    "@type" : "RecurrenceRule",
    "frequency": "yearly",
    "byMonth": ["1"],
    "byDay": [{
          "day": "su"
        },
        }
          "day": "mo"
        },
        }
          "day": "tu"
        },
        }
          "day": "we"
        },
        }
          "day": "th"
        },
        }
          "day": "fr"
        },
        }
          "day": "sa"
        }],
    "until": "2022-05-12T10:00:00"
  }],
  ...
```

```
...
RRULE:FREQ=MONTHLY;COUNT=6;BYDAY=-2MO
...
```

maps to

```
...
"recurrenceRules" : [{
    "@type" : "RecurrenceRule",
    "frequency": "monthly",
    "byDay": [{
        "day": "mo",
        "nthOfPeriod": -2
      }],
    "count": 6
  }],
  ...
```

5.32. SEQUENCE

Copy the value into the JSCalendar "sequence" property.

5.33. STATUS

For a VEVENT copy the lower-cased value into the JSCalendar "status" property.

For a VTODO copy the lower-cased value into the JSCalendar "progress" property.

5.34. STRUCTURED-DATA

This property is mapped on to a JSCalendar "link" object with the value mapped on to the JSCalendar "href" property in a manner depending on the "STRUCTURED-DATA" "VALUE" parameter:

VALUE=TEXT Copy the value as a [RFC2397] data uri either as plain text or by encoding as a base64 value. If plain text the value may need escaping as per [RFC2397].

VALUE=BINARY Copy the value as a [RFC2397] data uri speifying base64 encoding.

VALUE=URI Copy the value as-is into the href.

The "STRUCTURED-DATA" "SCHEMA" parameter is mapped on to a JSCalendar "schema" property within the link object.

The "STRUCTURED-DATA" "FMTTYPE" parameter is mapped on to a JSCalendar "contentType" property within the link object.

For example:

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

maps to (with data truncated)

```
...
"links": {
  "1": {
    "@type" : "Link",
    "contentType": "application/ld+json",
    "schema": "https://schema.org/SportsEvent",
    "href": "data:base64;ewogICAgICAgICJAY29udGV4dCI6IC..."
  }
}
...
```

5.35. SUMMARY

Copy the value into the JSCalendar "title" property.

Mapping parameters:

ALTREP No mapping.

LANG Use the "locale" property.

5.36. TRANSP

If the value of the TRANSP property (ignoring case) is "opaque" set the JSCalendar "freeBusyStatus" property to the value "busy".

Otherwise set the JSCalendar "freeBusyStatus" property to the value "free".

5.37. UID

Copy the value into the JSCalendar "uid" property.

5.38. URL

Maps to a Link object in the JSCalendar object's "links" property, with the URL property value mapped to the Link "href" property.

6. Translating iCalendar Recurrences

6.1. Translating iCalendar Recurrences: Simple objects with overrides

A simple object with overrides will be converted to a jsCalendar master event with the rules, recurrence dates and exclusion dates translated appropriately.

Overrides MUST be mapped on to a jsCalendar patch object and added to the "recurrenceOverrides" property of the master event with the key being the value of the iCalendar RECURRENCE-ID translated to a json format.

Any override property with the same value as the master SHOULD be ommitted. Remaining properties MAY be added in full. Where appropriate, differences SHOULD be expressed as a patch.

This can result in a significant reduction in size for objects with small changes to overrides, for example changing the participation status of an attendee.

6.2. Translating iCalendar Recurrences: Overrides with no master

When inviting an attendee to a single instance of a recurring event, only that override should be sent to the attendee. In this case the override should be a complete jsCalendar object with the type set to the type of the master.

Additionally, there MUST be a recurrenceId property set to the value of the recurrence id for that override. If the timezone of the start of the instance is different from the master value, then there must also be a "recurrenceIdTimeZone" property set to the start timezone of the master.

7. Translating iCalendar: Further examples

This section provides more complete examples of translating from [RFC5545] to JSCalendar.

As usual note that json string values may be split because of line width limits. This is not legal json.

7.1. Recurring event with ATTACH

This is an example of a recurring event with overrides. The first override removes an ATTACH property and adds an ATTACH property. The second override removes all ATTACH properties.

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//example.org//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200522T120000
DURATION:PT1H
RRULE:FREQ=DAILY;COUNT=8
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBDErecur-1
ATTACH;FMTTYPE=text/plain:http://example.org/doc1.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc2.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc3.txt
END:VEVENT
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200523T120000
DURATION:PT1H
RECURRENCE-ID;TZID=America/New_York:20200523T120000
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBDErecur-1
ATTACH;FMTTYPE=text/plain:http://example.org/doc2.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc3.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc4.txt
END:VEVENT
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200524T120000
DURATION:PT1H
RECURRENCE-ID;TZID=America/New_York:20200524T120000
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBDErecur-1
END:VEVENT
END:VCALENDAR
```

maps to

```
{
```

```
      "prodId": "//example.org//EN",
      "entries": [
        {
          "links": {
            "1": {
              "@type": "Link",
              "rel": "enclosure",
              "contentType": "text/plain",
              "href": "http://example.org/doc1.txt"
            },
            "2": {
              "@type": "Link",
              "rel": "enclosure",
              "contentType": "text/plain",
              "href": "http://example.org/doc2.txt"
            },
            "3": {
              "@type": "Link",
              "rel": "enclosure",
              "contentType": "text/plain",
              "href": "http://example.org/doc3.txt"
            }
          },
          "created ": "2020-05-23T17:04:50Z",
          "start": "2020-05-22T12:00:00",
          "timeZone": "America/New_York",
          "duration": "PT1H",
          "title": "recurring daily 8 times",
          "uid": "6252D6C40A8308BFE25BBDErecur-1",
          "recurrenceRules": [
            {
              "@type": "RecurrenceRule",
              "frequency": "daily",
              "count": 8
            }
          ],
          "recurrenceOverrides": {
            "2020-05-23T12:00:00": {
              "recurrenceId": "2020-05-23T12:00:00",
              "links/d4a618d4-929c-4c81-ae5b-322afe407a00": null,
              "links/fb75b76a-a159-4a86-bd3d-7ace6b39c6c3": {
                "@type": "Link",
                "rel": "enclosure",
                "contentType": "text/plain",
                "href": "http://example.org/doc4.txt"
              }
            },
            "2020-05-24T12:00:00": {
```

```
              "recurrenceId": "2020-05-24T12:00:00",
              "links/d4a618d4-929c-4c81-ae5b-322afe407a00": null,
              "links/6c54e72e-3413-487c-ae14-fb318a90db43": null,
              "links/44087e9a-132c-4a5d-b25d-4ce580edb004": null
            }
          }
        }
      ]
    }
```

7.2. Simple event with CONTACT

This example shows the conversion of a simple event with a single CONTACT property in JSCalendar.

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//Example//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200622T120000
DURATION:PT1H
SUMMARY:event with contact
UID:6252D6C40A8308BFE25BBEFcontact-1
CONTACT;ALTREP="ldap://example.com:6666/o=ABC%20Industries\,
 c=US???(cn=Jim%20Dolittle)":Jim Dolittle\, ABC Industries\,
 +1-919-555-1234
END:VEVENT
END:VCALENDAR
```

translates to

```
{
  "@type": "jsgroup",
  "prodId": "//Example.org//Example V3.13.2//EN",
  "entries": [
    {
      "@type": "jsevent",
      "participants": {
        "40288108-733187c1-0173-3188007b-00000001": {
          "@type": "Participant",
          "roles": {
            "contact": true
          },
          "description": "Jim Dolittle, ABC Industries,\
                          +1-919-555-1234",
          "links": {
```

```
                "1": {
                  "@type": "Link",
                  "href": "ldap://example.com:6666/o=ABC%20Industries,\
                                  c=US???(cn=Jim%20Dolittle)",
                  "rel": "alternate"
                }
              }
            }
          },
          "created ": "2020-07-09T03:04:23Z",
          "start": "2020-06-22T12:00:00",
          "timeZone": "America/New_York",
          "duration": "PT1H",
          "title": "event with contact",
          "uid": "6252D6C40A8308BFE25BBEFcontact-1"
        }
      ]
    }
```

7.3. Simple event with RESOURCES

TBD

7.4. Recurring event. Attendees only in overrides

In this more complex example there is no ORGANIZER or ATTENDEEs in the master event. There are overrides which invite one or more attendees.

For one overrride the ORGANIZER is also an ATTENDEE. In the other that is not the case. This is reflected in the "roles" property for the organizer.

Note that each override has its own "participants" property and the first has a links property to handle the DIR parameter on one attendee.

```
BEGIN:VCALENDAR
PRODID://Example.org//Example V3.13.2//EN
VERSION:2.0
BEGIN:VEVENT
CREATED:20200704T035515Z
DURATION:PT1H
DTSTAMP:20200704T035706Z
DTSTART;TZID=America/New_York:20200522T120000
LAST-MODIFIED:20200704T035706Z
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
```

```
RRULE:FREQ=DAILY;COUNT=8
END:VEVENT
BEGIN:VEVENT
RECURRENCE-ID;TZID=America/New_York:20200523T120000
ATTENDEE:mailto:douglm@example.org
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2;DIR="http://example.org/
 vcards/vbede.vcf":mailto:vbede@example.org
CREATED:20200704T035515Z
DURATION:PT1H
DTSTAMP:20200704T035706Z
DTSTART;TZID=America/New_York:20200523T120000
LAST-MODIFIED:20200704T035706Z
ORGANIZER:mailto:douglm@example.org
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
BEGIN:VEVENT
RECURRENCE-ID;TZID=America/New_York:20200524T120000
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2:mailto:user01@example.org
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2:mailto:vbede@example.org
CREATED:20200704T035515Z
DURATION:PT1H
DTSTAMP:20200704T035706Z
DTSTART;TZID=America/New_York:20200524T120000
LAST-MODIFIED:20200704T035706Z
ORGANIZER:mailto:douglm@example.org
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
END:VCALENDAR
```

translates to

```
{
  "@type": "jsgroup",
  "prodId": "//Example.org//Example V3.13.2//EN",
  "entries": [
    {
      "@type": "jsevent",
      "created ": "2020-07-04T03:57:06Z",
      "start": "2020-05-22T12:00:00",
      "timeZone": "America/New_York",
      "duration": "PT1H",
      "title": "recurring daily 8 times",
      "uid": "6252D6C40A8308BFE25BBEFrecur1-1",
      "recurrenceRules": [
        {
          "@type": "RecurrenceRule",
```

```
      "frequency": "daily",
      "count": 8
    }
  ],
  "recurrenceOverrides": {
    "2020-05-23T12:00:00": {
      "participants": {
        "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
          "@type": "Participant",
          "sendTo": {
            "imip": "mailto:douglm@example.org"
          },
          "roles": {
            "attendee": true,
            "owner": true
          }
        },
        "a539dfe3-4463-4f28-b9de-17d3a0e99faf": {
          "@type": "Participant",
          "sendTo": {
            "imip": "mailto:vbede@example.org"
          },
          "expectReply": true,
          "links": {
            "1": {
              "@type": "Link",
              "href": "http://example.org/vcards/vbede.vcf",
              "rel": "alternate"
            }
          },
          "roles": {
            "attendee": true
          },
          "scheduleStatus": "1.2"
        }
      },
      "replyTo": {
        "imip": "mailto:douglm@example.org"
      }
    },
    "2020-05-24T12:00:00": {
      "participants": {
        "daeae4cf-6f6a-4ce3-9f4d-6bd884650d3d": {
          "@type": "Participant",
          "sendTo": {
            "imip": "mailto:user01@example.org"
          },
          "expectReply": true,
```

```
                    "roles": {
                      "attendee": true
                    },
                    "scheduleStatus": "1.2"
                  },
                  "a6de6de3-271f-4679-9241-1b3bca6b602d": {
                    "@type": "Participant",
                    "sendTo": {
                      "imip": "mailto:vbede@example.org"
                    },
                    "expectReply": true,
                    "roles": {
                      "attendee": true
                    },
                    "scheduleStatus": "1.2"
                  },
                  "aaa8483b-b18b-4dbd-b218-77d8db027d35": {
                    "@type": "Participant",
                    "sendTo": {
                      "imip": "mailto:douglm@example.org"
                    },
                    "roles": {
                      "owner": true
                    }
                  }
                },
                "replyTo": {
                  "imip": "mailto:douglm@example.org"
                }
              }
            }
          }
        ]
      }
```

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

[draft-ietf-calext-eventpub-extensions]
"Event Publishing Extensions to iCalendar", <https://tools.ietf.org/html/draft-ietf-calext-eventpub-extensions>.

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

[draft-ietf-calext-valarm-extensions]
"VALARM Extensions for iCalendar", <https://tools.ietf.org/html/draft-ietf-calext-valarm-extensions>.

[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>.

[RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, DOI 10.17487/RFC2397, August 1998, <https://www.rfc-editor.org/info/rfc2397>.

[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>.

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

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

Appendix A. Outdated document sections

A.1. Translating JSCalendar properties to iCalendar components

This section is an alphabetic list of all JSCalendar property types that map on to components.

A.1.1. jsevent

A JSCalendar object with a type of "jsevent" is mapped on to a [RFC5545] VEVENT component.

If it is a single VEVENT then a [RFC5545] VCALENDAR component must surround it and the JSCalendar "prodid" property will be converted to a [RFC5545] PRODID.

```
{
  "@type": "jsevent",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
}
```

maps to

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
BEGIN:VEVENT
...
END:VEVENT
END: VCALENDAR
```

When converting multiple JSEvent or JSTask objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the JSGroup "prodid" or generated.

A.1.2. jsgroup

A JSCalendar object with a type of "jsgroup" is mapped on to a [RFC5545] VCALENDAR component.

```
{
  "@type": "jsgroup",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
  {
    "@type": "jsevent",
    ...
  }
  {
    "@type": "jsevent",
    ...
  }
}
```

maps to

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
BEGIN:VEVENT
...
END:VEVENT
BEGIN:VEVENT
...
END:VEVENT
END: VCALENDAR
```

A.1.3. jstask

A JSCalendar object with a type of "jstask" is mapped on to a [RFC5545] VTODO component.

If it is a single VTODO then a [RFC5545] VCALENDAR component must surround it and the JSCalendar "prodid" property will be converted to a [RFC5545] PRODID.

```
{
  "@type": "jstask",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
}
```

maps to

```
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
BEGIN:VTODO
...
END:VTODO
END: VCALENDAR
```

When converting multiple JSEvent or JSTask objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the JSGroup "prodid" or generated.

A.1.4. timezones

The JSCalendar TimeZone objects within a "timezones" property are mapped on to [RFC5545] VTIMEZONE components within the surrounding VCALENDAR component. Each mapped TimeZone MUST only appear once.

```
{
  "@type": "jsevent",
  ...
  "timezones": {
    "/Example/Somewhere: {
      "@type": "TimeZone",
      ...
    },
    "/Example/Somewhere-else": {
      "@type": "TimeZone",
      ...
    }
  }
}
```

maps to

```
BEGIN: VTIMEZONE
TZID: /Example/Somewhere
...
END: VTIMEZONE
BEGIN: VTIMEZONE
TZID: /Example/Somewhere-else
...
END: VTIMEZONE
BEGIN: VEVENT
...
END: VEVENT
```

When converting multiple JSEvent or JSTask objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the JSGroup "prodid" or generated.

A.1.5. locations

JSCalendar locations should be mapped on to [draft-ietf-calext-eventpub-extensions]VLOCATION components. Additionally, for backwards compatibility, a location should be mapped on to a [RFC5545] LOCATION property. This property should be mapped from the only location or the one related to the start.

A.1.6. participants

JSCalendar participants will be mapped on to different iCalendar properties and components depending on their jsCalendar role values.

A participant with a role containing "contact" MUST be mapped on to an iCalendar CONTACT property and SHOULD also be mapped on to a

[draft-ietf-calext-eventpub-extensions]PARTICIPANT component which provides a better mapping.

A participant with a role containing "owner" MUST be mapped on to an iCalendar ORGANIZER property and SHOULD also be mapped on to a [draft-ietf-calext-eventpub-extensions]PARTICIPANT component which provides a better mapping.

A participant with a role containing any of "attendee", "optional" or "informational" MUST be mapped on to an iCalendar ATTENDEE property and SHOULD also be mapped on to a [draft-ietf-calext-eventpub-extensions]PARTICIPANT component which provides a better mapping.

A more complete mapping may be achieved by creating a [draft-ietf-calext-eventpub-extensions]PARTICIPANT component.

For all properties the participants jsCalendar "language" property, if present, is mapped on to the iCalendar "LANG" property parameter.

For all properties if the participant contains a jsCalendar "link" with a "rel" of "alternate" then the value of the link is used for the iCalendar "ALTREP" property parameter.

Where do we get the cua?

A.2. Note

The sections following this one are all the original ones from draft 1 writen by Robert/Neil - there for reference.

A.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

A.4. JSTask

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

Property	iCalendar counterpart
due	Maps to the DUE property. See Appendix A.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

A.5. JSGroup

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

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

Table 3: Mapping JSGroup properties

A.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 VTODO, "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 Appendix A.6.2.
method	METHOD property of the embedding VCALENDAR.
participants	See Appendix A.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 _until_ 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 Appendix A.6.1.
status	STATUS property.
timeZone	Maps to the TZID parameter. See Appendix A.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 Appendix A.6.2.

Table 4: Translation between JSCalendar and iCalendar

A.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:

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

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

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

Fractional seconds SHOULD be preserved with the SUBSECOND parameter.

A.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.
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

A.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
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

A.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.

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

Michael Douglass
Bedework Commercial Services
226 3rd Street
Troy, NY 12180
United States of America

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

Network Working Group
Internet-Draft
Updates: 5988 (if approved)
Intended status: Standards Track
Expires: 5 August 2021

M. Douglass

1 February 2021

Calendar subscription upgrades
draft-ietf-calext-subscription-upgrade-03

Abstract

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 5 August 2021.

Copyright Notice

Copyright (c) 2021 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. 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.

2. 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 perfom 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. Additionally an enhanced GET protocol is specified here to allow a light weight implementation.

The use of subscription upgtafe may help reduce load on servers, but perhaps more inportantly it allows mobile devices to use a more efficient update mechanism reducing data tranferred and presumably improving battery life.

2.1. 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].

3. 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.

EXAMPLE

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"
```

Figure 1

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

4. Enhanced GET

4.1. 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.

4.2. 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.

4.3. 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.

4.4. 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.

4.5. 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:

* subscribe-enhanced-get

* limit

4.6. Examples

EXAMPLE 1

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
```

Figure 2

EXAMPLE 2

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
```

Figure 3

EXAMPLE 3

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
```

Figure 4

EXAMPLE 4

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
```

Figure 5

5. Changes to the iCalendar specifications

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

5.1. 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".
```

Figure 6

Example

EXAMPLE 1

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

```
STATUS:TENTATIVE
```

Figure 7

EXAMPLE 2

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

STATUS:NEEDS-ACTION

Figure 8

EXAMPLE 3

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

STATUS:DRAFT

Figure 9

6. Header Field: Sync-Token

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

```
Accept = DQUOTE URI DQUOTE
```

Figure 10

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.

EXAMPLE

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

```
Sync-Token: "data:,1234567"
```

Figure 11

7. New Prefer header field preferences

7.1. 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"
```

Figure 12

7.2. Preference limit

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

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

Figure 13

8. Link relations

8.1. General

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

8.2. 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.

8.3. 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.

8.4. 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.

8.5. subscribe-enhanced-get

This specifies an access point which supports something new.

The client MUST follow that specification.

9. 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].

10. IANA Considerations

10.1. 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>

Figure 14

10.2. 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

10.3. Link Relation Registrations

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

Relation Name	Description	Reference
subscribe-caldav	Current	Section 8.2
subscribe-caldav_auth	Current	Section 8.3
subscribe-webdav-sync	Current	Section 8.4
subscribe-enhanced_get	Current	Section 8.5

Table 1

11. Normative references

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

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", IETF RFC 3864, IETF 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", IETF RFC 3986, IETF RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

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

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", IETF RFC 8174, IETF 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

* First calext version

* Use Sync-Token header rather than parameter

v04 2019-03-07 MD

* Reference to RFC 6538 - WebDAV sync and RFC 7240 Prefer

* Go back to HEAD

* New Preference and parameters.

* Examples

* More text for extended get. Talk about deletions.

v01 2017-02-17 MD

* Add text about OPTIONS

* Add text abut read/write CalDAV

v00 2017-02-15 MD

* First pass

Author's Address

Michael Douglass

Email: mdouglass@bedework.com

Network Working Group
Internet-Draft
Updates: 5545 (if approved)
Intended status: Standards Track
Expires: August 19, 2021

C. Daboo
Apple
K. Murchison, Ed.
Fastmail

February 15, 2021

VALARM Extensions for iCalendar
draft-ietf-calext-valarm-extensions-05

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.

This document updates RFC5545.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 19, 2021.

Copyright Notice

Copyright (c) 2021 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

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 prevalent, 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 [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 are OPTIONAL,
                ; and MAY occur more than once

                x-prop / iana-prop

                )

   actionprops = *audiopropext / *dispropext / *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

                   )

   dispropext = (

                  ; 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

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

                attach

                )
```

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 Related To

It is often convenient to relate one or more "VALARM" components to other "VALARM" components (e.g., see Section 7). This can be accomplished if the "VALARM" components each have their own "UID" property (as per Section 4).

This specification updates the usage of the "RELATED-TO" property defined in Section 3.8.4.5 of [RFC5545] to enable its use with "VALARM" components. Specific types of relationships between "VALARM" components can be identified by registering new values for the "RELTYPE" property parameter defined in Section 3.2.15 of [RFC5545].

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

```
alarmprop  =/ (

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

              related

              )
```

6. 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

                   )
```

6.1. 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 alarms, such as email-based alerts, might not provide feedback as to when the calendar user sees them. 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
```

7. 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.

Clients MUST add a "RELATED-TO" property to the new "VALARM" component with a value set to the "UID" property value of the "VALARM" component being snoozed. If the "VALARM" component being snoozed does not already have a "UID" property, the client MUST add one. The "RELATED-TO" property added to the new "VALARM" component MUST include a "RELTYPE" property parameter with a value set to "SNOOZE".

When the "snooze" alarm is triggered and dismissed the client MUST do one of the following with the corresponding "VALARM" component:

- Set the "ACKNOWLEDGED" property (see Section 6.1)
- Remove the component

Alternatively, if the "snooze" alarm is itself "snoozed", the client MUST remove the original "snooze" alarm and create a new one, with the appropriate trigger time and relationship set.

Note that regardless of the final disposition of the "snooze" alarm when triggered, the original "VALARM" component is left unchanged.

7.1. Relationship Type Property Parameter

This specification adds the "SNOOZE" relationship type for use with the "RELTYPE" property defined in Section 3.2.15 of [RFC5545]. This is used when relating a "snoozed" "VALARM" component to the original alarm that the "snooze" was generated for.

8. Alarm Proximity Trigger

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

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

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

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

```
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.

8.1. 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 MUST be 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. Note that the meaning of "vicinity" in this context is implementation defined.

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

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
```

8.2. 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
```

9. Security Considerations

In addition to the security properties of iCalendar (see Section 7 of [RFC5545]), 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.

10. Privacy Considerations

Proximity VALARMs, if not used carefully, can leak a user's past, present, or future location. For instance, storing an iCalendar resource containing proxmity VALARMs to a shared calendar on CalDAV server can expose to anyone that has access to that calendar the user's intent to leave from or arrive at a particular location at some future time. Furthermore, if a CalDAV client updates the shared iCalendar resource with an ACKNOWLEDGED property when the alarm is triggered, will leak the exact date and time that the user left from or arrived at the location. Therefore, CalDAV clients that implement proximity alarms SHOULD give users the option of storing and/or acknowledging the alarms on the local device only and not storing the alarm and/or acknowledgment on a remote server.

11. IANA Considerations

11.1. Property Registrations

This document defines the following new iCalendar properties to be added to the registry defined in Section 8.2.3 of [RFC5545] and located here: <https://www.iana.org/assignments/icalendar#properties>

Property	Status	Reference
ACKNOWLEDGED	Current	RFCXXXX, Section 6.1
PROXIMITY	Current	RFCXXXX, Section 8.1

11.2. Relationship Types Registry

This document defines the following new iCalendar relationship type to be added to the registry defined in Section 8.3.8 of [RFC5545] and located here: <https://www.iana.org/assignments/icalendar#relationship-types>

Relationship Type	Status	Reference
SNOOZE	Current	RFCXXXX, Section 7.1

11.3. Proximity Value Registry

This document creates a new iCalendar registry for values of the "PROXIMITY" property located here: <https://www.iana.org/assignments/icalendar#proximity-values>

Additional values MAY be used, provided the process described in Section 8.2.1 of [RFC5545] is used to register them, using the template in Section 8.2.6 of [RFC5545].

The following table has been used to initialize the Proximity Value Registry.

Value	Status	Reference
ARRIVE	Current	RFCXXXX, Section 8.1
DEPART	Current	RFCXXXX, Section 8.1
CONNECT	Current	RFCXXXX, Section 8.1
DISCONNECT	Current	RFCXXXX, Section 8.1

12. 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, Ciny Joy, Barry Leiba, and Daniel Migault.

13. References

13.1. Normative References

[I-D.ietf-calext-eventpub-extensions]
Douglass, M., "Event Publishing Extensions to iCalendar", draft-ietf-calext-eventpub-extensions-18 (work in progress), January 2021.

[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>.

13.2. 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>.

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

Changes in ietf-04:

1. Addressed security review comments from Valery Smyslov.

2. Addressed Genart review comments from Roni Even.

3. Added text addressing management of Proximity Value Registry.

Changes in ietf-03:

1. Fixed ABNF to be properly extended.

2. Addressed AD review comments from Barry Leiba.

Changes in ietf-02:

1. Addressed some WGLC comments from Daniel Migault.

Changes in ietf-01:

1. Reintroduced the RELATED-TO property for VALARMs and the SNOOZE value for the RELTYPE property parameter.

2. Add Privacy Considerations section.

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
Philadephia, PA 19102
USA

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 April 2021

E. York

C. Daboo

M. Douglass

14 October 2020

VPOLL: Consensus Scheduling Component for iCalendar
draft-ietf-calext-vpoll-01

Abstract

This specification introduces a new iCalendar component which allows for consensus scheduling, that is, voting on a number of alternative meeting or task alternatives.

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 17 April 2021.

Copyright Notice

Copyright (c) 2020 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. Acknowledgements

The authors would like to thank the members of the Calendaring and Scheduling Consortium (CalConnect) for contributing their ideas and support.

2. Introduction

The currently existing approach to agreeing on meeting times using iTip [RFC5546] and/or iMip [RFC6047] has some significant failings. There is no useful bargaining or suggestion mechanism in iTip, only the ability for a potential attendee to accept or refuse or to counter with a time of their own choosing.

Part of the problem is that for many potential attendees, their freebusy is not an accurate representation of their availability. In fact, when trying to schedule conference calls across different organizations, attendees may not be allowed to provide freebusy information or availability as this may reveal something of the organizations internal activities.

A number of studies have shown that large amounts of time are spent trying to come to an agreement - up to and beyond 20 working hours per meeting. Many organizers fall back on other approaches such as phone calls and email to determine a suitable time.

Online services have appeared as a result and these allow participants to vote on a number of alternatives without revealing or using freebusy or availability. When agreement is reached a conventional scheduling message may be sent to the attendees. This approach appears to reach consensus fairly rapidly. Peer pressure may have some bearing on this as all voters are usually able to see the current state of the voting and may adjust their own meeting schedules to make themselves available for a popular choice.

The component and properties defined in this specification provide a standardized structure for this process and allow calendar clients and servers and web based services to interact.

These structures also have uses beyond the relatively simple needs of most meeting organizers. The process of coming to consensus can also be viewed as a bidding process.

3. Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1. consensus scheduling

The process whereby users come to some agreement on meeting or task alternatives and then book that meeting or task.

3.2. active Vpoll

A VPoll may have a DTSTART, DTEND and DURATION which may define the start and end of the active voting period

3.3. voter

A participant who votes on the alternatives. A voter need not be an attendee of any of the alternatives presented.

4. Simple Consensus Scheduling

This specification defines components and properties which can be used for simple consensus scheduling but also have the generality to handle more complex cases. To provide an easy (and for many a sufficient) introduction to consensus scheduling and VPOLL we will outline the flow of information for the simple case of voting on a number of meeting alternatives which differ only in time of the meeting. In addition the voters will all be potential attendees.

This specification not only defines data structures but adds new iTip methods, one used when consensus has been reached and one to distribute the current status of the poll.

This document will show how a VPOLL object is used to inform voters of the state of a simple vote on some alternatives.

4.1. The VPOLL Component: An Overview

The VPOLL component acts as a wrapper for a number of alternatives to be voted on, together with some properties and a new component used to maintain the state of the voting. For our simple example the following VPOLL properties and sub-components are either required or appropriate:

DTSTAMP The usual [RFC5545] property.

SEQUENCE The usual [RFC5545] property. See below for SEQUENCE behavior.

UID The usual [RFC5545] property.

ORGANIZER The usual [RFC5545] property. In general this need not be an organizer of any of the alternatives. In this simple outline we assume it is the same.

SUMMARY The usual [RFC5545] property. This optional but recommended property provides the a short title to the poll.

DESCRIPTION The usual [RFC5545] property. This optional property provides more details.

DTEND The usual [RFC5545] property. This optional property provides a poll closing time and date after which the VPOLL is no longer active.

POLL-MODE A new property which defines how the votes are used to

obtain a result. For our use case it will take the value "BASIC" meaning one event will be chosen from the alternatives.

POLL-COMPLETION A new property which defines who (server or client) chooses and/or submits the winning choice. In our example the value is "SERVER-SUBMIT" which means the client chooses the winner but the server will submit the winning choice.

POLL-PROPERTIES A new property which defines which icalendar properties are being voted on. For our use case it will take the value "DTSTART, LOCATION" meaning only those properties are significant for voting. Other properties in the events may differ but are not considered significant for the voting process.

PARTICIPANT There is one of these components for each voter with the PARTICIPANT-TYPE set to "VOTER". The CALENDAR-ADDRESS property identifies the voter and this component will contain one VOTE component for each item being voted on.

VOTE A new component. There is one of these for each voter and choice. It usually contains at least a POLL-ITEM-ID property to identify the choice and a RESPONSE property to provide a vote. For more complex poll modes it may contain other information such as cost or estimated duration.

VEVENT In our simple use case there will be multiple VEVENT sub-components defining the alternatives. Each will have a different date and or time for the meeting.

EXAMPLE

VPOLL with 3 voters and 3 alternative meetings:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD:REQUEST
BEGIN:VPOLL
POLL-MODE:BASIC
POLL-COMPLETION:SERVER-SUBMIT
POLL-PROPERTIES:DTSTART,LOCATION
ORGANIZER:mailto:mike@example.com
UID:sched01-1234567890
DTSTAMP:20120101T000000Z
SUMMARY:What to do this week
DTEND:20120101T000000Z
BEGIN: PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:cyrus@example.com
END PARTICIPANT
BEGIN: PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:eric@example.com
END PARTICIPANT
BEGIN: PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:mike@example.com
END PARTICIPANT
BEGIN:VEVENT.......(with a poll-item-id=1)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=2)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=3)
END:VEVENT
END:VPOLL
END:VCALENDAR
```

Figure 1

As can be seen in the example above, there is an iTip METHOD property with the value REQUEST. The VPOLL object will be distributed to all the voters, either through iMip or through some VPOLL enabled service.

4.2. The VPOLL Alternative Choices: An Overview

Within the VPOLL component we have the alternatives to vote on. In many respects these are standard [RFC5545] components. For our simple use case they are all VEVENT components. In addition to the usual [RFC5545] properties some extra properties are used for a VPOLL.

POLL-ITEM-ID This provides a unique reference to the sub-component within the VPOLL. It's value SHOULD be a small integer.

4.3. VPOLL responses

Upon receipt of a VPOLL REQUEST the voter will reply with a VPOLL component containing their vote. In our simple case it will have the following properties and components:

DTSTAMP The usual [RFC5545] property.

SEQUENCE The usual [RFC5545] property. See below for SEQUENCE behavior.

UID Same as the request.

ORGANIZER Same as the request.

SUMMARY Same as the request.

PARTICIPANT One only with a CALENDAR-ADDRESS identifying the voter replying.

VOTE One per item being voted on.

POLL-ITEM-ID One inside each VOTE component to identify the choice.

RESPONSE One inside each VOTE component to specify the vote.

Note that a voter can send a number of REPLYs for each REQUEST sent by the organizer. Each REPLY completely replaces the voting record for that voter for all components being voted on. In our example, if Eric responds and votes for items 1 and 2 and then responds again with a vote for only item 3, the final outcome is one vote on item 3.

NOTE This is poll-mode specific behavior?

EXAMPLE

REPLY VPOLL from Cyrus:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD: REPLY
BEGIN:VPOLL
ORGANIZER:mailto:mike@example.com
UID:sched01-1234567890
DTSTAMP:20120101T010000Z
SUMMARY:What to do this week
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:cyrus@example.com
BEGIN:VOTE
POLL-ITEM-ID:1
RESPONSE:50
COMMENT:Work on iTIP
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:2
RESPONSE:100
COMMENT:Work on WebDAV
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:3
RESPONSE:0
END:VOTE
END:PARTICIPANT
END:VPOLL
END:VCALENDAR
```

Figure 2

4.4. VPOLL updates

When the organizer receives a response from one or more voters the current state of the poll is sent to all voters. The new iTip method POLLSTATUS is used. The VPOLL can contain a reduced set of properties but MUST contain DTSTAMP, SEQUENCE (if not 0), UID, ORGANIZER and one or more PARTICIPANT components each populated with zero or more VOTE components.

EXAMPLE

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD: POLLSTATUS
BEGIN:VPOLL
ORGANIZER:mailto:mike@example.com
UID:sched01-1234567890
DTSTAMP:20120101T020000Z
SEQUENCE:0
SUMMARY:What to do this week
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:cyrus@example.com
BEGIN: VOTE
POLL-ITEM-ID:1
RESPONSE:50
COMMENT:Work on iTIP
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:2
RESPONSE:100
COMMENT:Work on WebDAV
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:3
RESPONSE:0
END:VOTE
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:eric@example.com
BEGIN:VOTE
POLL-ITEM-ID:1
RESPONSE:100
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:2
RESPONSE:100
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:3
RESPONSE:0
END:VOTE
END:PARTICIPANT
END:VPOLL
END:VCALENDAR
```

Figure 3

4.5. VPOLL Completion

After a number of REPLY messages have been received the poll will be considered complete. If there is a DTEND on the poll the system may automatically close the poll, or the organizer may, at any time, consider the poll complete. A VPOLL can be completed (and effectively closed for voting) by sending an iTip REQUEST message with the VPOLL STATUS property set to COMPLETED.

The poll winner is confirmed by sending a final iTip REQUEST message with the VPOLL STATUS property set to CONFIRMED. In this case the VPOLL component contains all the events being voted on along with a POLL-WINNER property to identify the winning event. As the POLL-COMPLETION property is set to SERVER-SUBMIT the server will submit the winning choice and when it has done so set the STATUS to "SUBMITTED".

EXAMPLE

VPOLL confirmation:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD: REQUEST
BEGIN:VPOLL
ORGANIZER:mailto:douglm@example.com
UID:sched01-1234567890
DTSTAMP:20120101T030000Z
COMPLETED:20120101T030000Z
POLL-COMPLETION:SERVER-SUBMIT
SEQUENCE:0
SUMMARY:What to do this week
STATUS:CONFIRMED
POLL-WINNER:3
BEGIN:VEVENT.......(with a poll-item-id=1)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=2)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=3)
END:VEVENT
END:VPOLL
END:VCALENDAR
```

Figure 4

4.6. Other Responses

A voter being asked to choose between a number of ORGANIZER supplied alternatives may find none of them acceptable or may simply not care.

An alternative response, which may be disallowed by the ORGANIZER, is to send back the respondees availability or freebusy or even one or more new, alternative choices.

This is accomplished by responding with a VOTE component which has no POLL-ITEM-ID property. In this case it MUST contain some alternative information. What form this takes depends on the poll mode in effect.

5. iCalendar Extensions

5.1. Updated Participant Type Value

Participant type property values are defined in section 11.2.1. of [I-D.ietf-calext-eventpub-extensions]. This specification updates that type to include the new participant type VOTER to provide information about the voter and to contain their votes.

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

```
partvalue       /= "VOTER"
```

Figure 5

Description The new property value indicates that the associated PARTICIPANT component identifies a voter in a VPOLL.

5.2. Updated Relation Type Value

Relationship parameter type values are defined in section 3.2.15. of [RFC5545]. This specification updates that type to include the new relationship value POLL to provide a link to the VPOLL component in which the current component appears.

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

```
reltypeparam       /= "RELTYPE" "=" "POLL"
; Property value is a VPOLL uid
```

Figure 6

Description This parameter can be specified on a property that references another related calendar component. The new parameter value indicates that the associated property references a VPOLL component which contains the current component.

5.3. Updated Status Value

Status property values are defined in section 3.8.1.11. of [RFC5545]. This specification updates that type to define valid VPOLL status values.

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

```
statvalue /= statvalue-poll
   ; Status values for "VPOLL".
statvalue-poll = "IN-PROCESS"
          / "COMPLETED"  ; Poll has closed,
                         ; nothing has been chosen yet
          / "CONFIRMED"  ; Poll has closed and
                         ; winning items confirmed
          / "SUBMITTED"  ; The winning item has been
                         ; submitted
          / "CANCELLED"
```

Figure 7

Description These values allow clients and servers to handle the choosing and submission of winning choices.

If the client is choosing and the server submitting then the client should set the POLL-WINNER property, set the status to CONFIRMED and save the poll. When the server submits the winning choice it will set the status to SUBMITTED.

Figure 8

5.4. New Property Parameters

5.4.1. Required

Parameter name REQUIRED

Purpose To specify whether the associated property is required in the current context.

Format Definition This parameter is defined by the following notation:

```
requirededparam = "REQUIRED"  "=" ("TRUE" / "FALSE")
  ; Default is FALSE
```

Figure 9

Description This parameter MAY be specified on REPLY-URL and, if the value is TRUE, indicates the organizer requires all replies to be made via the specified service rather than iTip replies.

5.4.2. Stay-Informed

Parameter name STAY-INFORMED

Purpose To specify the voter also wants to be added as an ATTENDEE when the poll is confirmed.

Format Definition This parameter is defined by the following notation:

```
stayinformedparam = "STAY-INFORMED"  "=" ("TRUE" / "FALSE")
                  ; Default is FALSE
```

Figure 10

Description This parameter MAY be specified on the CALENDAR-ADDRESS property in the PARTICIPANT component and, if the value is TRUE, indicates the voter wishes to be added to the final choice as a non participant.

5.5. New Properties

5.5.1. Accept-Response

Property name ACCEPT-RESPONSE

Purpose This property is used in VPOLL to indicate the types of component that may be supplied in a response.

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

Conformance This property MAY be specified in a VPOLL component.

Description When used in a VPOLL this property indicates what allowable component types may be returned in a reply. Typically this would allow a voter to respond with their freebusy or availability rather than choosing one of the presented alternatives.

If this property is not present voters are only allowed to respond to the choices in the request.

Format Definition This property is defined by the following notation:

```
acceptresponse = "ACCEPT-RESPONSE" acceptresponseparams ":"
                    iana-token ("," iana-token) CRLF

acceptresponseparams = *(";" other-param)
```

Figure 11

5.5.2. Poll-Completion

Property name POLL-COMPLETION

Purpose This property is used in VPOLL to indicate whether the client or server is responsible for choosing and/or submitting the winner(s).

Description When a VPOLL is stored on a server which is capable of handling choosing and submission of winning choices a value of SERVER indicates that the server should close the poll, choose the winner and submit whenever it is appropriate to do so.

For example, in BASIC poll-mode, reaching the DTEND of the poll could trigger this server side action.

Server initiated submission requires that the submitted choice MUST be a valid calendaring component.

POLL-COMPLETION=SERVER-SUBMIT allows the client to set the poll-winner, set the status to CONFIRMED and then store the poll on the server. The server will then submit the winning choice and set the status to SUBMITTED.

Format Definition This property is defined by the following notation:

```
poll-completion = "POLL-COMPLETION" pcparam ":" pcvalue CRLF

pcparam = *(";" other-param)

pcvalue = "SERVER"  ; The server is responsible for both choosing and
                    ; submitting the winner(s)
        / "SERVER-SUBMIT" ; The server is responsible for
                    ; submitting the winner(s). The client chooses.
        / "SERVER-CHOICE"  ; The server is responsible for
                    ; choosing the winner(s). The client will submit.
        / "CLIENT" ; The client is responsible for both choosing and
                    ; submitting the winner(s)
        / iana-token
        / x-name
        ;Default is CLIENT
```

Figure 12

Example The following is an example of this property:

```
POLL-COMPLETION: SERVER-SUBMIT
```

Figure 13

5.5.3. Poll-Item-Id

Property name POLL-ITEM-ID

Purpose This property is used in VPOLL child components as an identifier.

Value type INTEGER

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

Conformance This property MUST be specified in a VOTE component and in VPOLL choice items.

Description In a METHOD:REQUEST each choice component MUST have a POLL-ITEM-ID property. Each set of components with the same POLL-ITEM-ID value represents one overall set of items to be voted on.

POLL-ITEM-ID SHOULD be a unique small integer for each component or set of components. If it remains the same between REQUESTs then the previous response for that component MAY be re-used. To force a re-vote on a component due to a significant change, the POLL-ITEM-ID MUST change.

Format Definition This property is defined by the following notation:

```
pollitemid = "POLL-ITEM-ID" pollitemdparams ":"
                  integer CRLF

pollitemidparams = *(
                    (";" other-param)
            )
```

Figure 14

5.5.4. Poll-Mode

Property name POLL-MODE

Purpose This property is used in VPOLL to indicate what voting mode is to be applied.

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

Conformance This property MAY be specified in a VPOLL component or its sub-components.

Description The poll mode defines how the votes are applied to obtain a result. BASIC mode, the default, means that the voters are selecting one component (or group of components) with a given POLL=ITEM-ID.

Other polling modes may be defined in updates to this specification. These may allow for such modes as ranking or task assignment.

Format Definition This property is defined by the following notation:

```
pollmode = "POLL-MODE" pollmodeparams ":"
             ("BASIC" / iana-token / other-token) CRLF

pollmodeparams = *(";" other-param)
```

Figure 15

5.5.5. Poll-properties

Property name POLL-PROPERTIES

Purpose This property is used in VPOLL to define which icalendar properties are being voted on.

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

Conformance This property MAY be specified in a VPOLL component.

Description This property defines which icalendar properties are significant in the voting process. It may not be clear to voters which properties are varying in a significant manner. Clients may use this property to highlight those listed properties.

Format Definition This property is defined by the following notation:

```
pollproperties = "POLL-PROPERTIES" pollpropparams ":"
             text *("," text) CRLF

pollpropparams = *(";" other-param)
```

Figure 16

5.5.6. Poll-Winner

Property name POLL-WINNER

Purpose This property is used in a basic mode VPOLL to indicate which of the VPOLL sub-components won.

Value type INTEGER

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

Conformance This property MAY be specified in a VPOLL component.

Description For poll confirmation each child component MUST have a POLL-ITEM-ID property. For basic mode the VPOLL component SHOULD have a POLL-WINNER property which MUST correspond to one of the POLL-ITEM-ID properties and indicates which sub-component was the winner.

Format Definition This property is defined by the following notation:

```
pollwinner = "POLL-WINNER" pollwinnerparams ":"
                integer CRLF

pollwinnerparams = *(";" other-param)

       ; Used with a STATUS:CONFIRMED VPOLL to indicate which
       ; components have been confirmed
```

Figure 17

5.5.7. Reply-URL

Property name REPLY-URL

Purpose This property may be used in scheduling messages to indicate additional reply methods, for example a web-service.

Property Parameters Non-standard, required or iana parameters can be specified on this property.

Conformance This property MAY be specified in a VPOLL component.

Description When used in a scheduling message this property indicates additional or required services that can be used to reply. Typically this would be a web service of some form.

Format Definition This property is defined by the following notation:

```
reply-url = "REPLY-URL" reply-urlparams ":" uri CRLF

reply-urlparams = *(
                  (";" requiredparam) /
                  (";" other-param)
                  )
```

Figure 18

5.5.8. Response

Property name RESPONSE

Purpose To specify a response vote.

Value type INTEGER

Format Definition This property is defined by the following notation:

```
response = "RESPONSE" response-params ":" integer CRLF
                ; integer value 0..100

responseparams = *(";" other-param)
```

Figure 19

Description This parameter can be specified on the POLL-ITEM-ID property to provide the value of the voters response. This parameter allows for fine grained responses which are appropriate to some applications. For the case of individuals voting for a choice of events, client applications SHOULD conform to the following convention:

* 0 - 39 A "NO vote"

* 40 - 79 A "MAYBE" vote

* 80 - 89 A "YES - but not preferred vote"

* 90-100 A "YES" vote.

Clients MUST preserve the response value when there is no change from the user even if they have a UI with fixed states (e.g. yes/no/maybe).

5.6. New Components

5.6.1. VPOLL Component

Component name VPOLL

Purpose This component provides a mechanism by which voters can vote on provided choices.

Format Definition This property is defined by the following notation:

```
   pollc    = "BEGIN" ":" "VPOLL" CRLF
              pollprop
              *participantc *eventc *todoc *journalc *freebusyc
              *availabilityc *alarmc *iana-comp *x-comp
              "END" ":" "VPOLL" CRLF

   pollprop = *(
             ;
             ; The following are REQUIRED,
             ; but MUST NOT occur more than once.
             ;
             dtstamp / uid / organizer /
             ;
             ; The following are OPTIONAL,
             ; but MUST NOT occur more than once.
             ;
             acceptresponse / class / created / completed /
             description / dtstart / last-mod / pollmode /
             pollproperties / priority / seq / status /
             summary / url /
             ;
             ; Either 'dtend' or 'duration' MAY appear in
             ; a 'pollprop', but 'dtend' and 'duration'
             ; MUST NOT occur in the same 'pollprop'.
             ; 'duration' MUST only occur when 'dtstart'
             ; is present
             ;
             dtend / duration /
             ;
             ; The following are OPTIONAL,
             ; and MAY occur more than once.
             ;
             attach / categories / comment /
             contact / rstatus / related /
             resources / x-prop / iana-prop
             ;
             ; The following is OPTIONAL, it SHOULD appear
             ; once for the confirmation of a BASIC mode
             ; VPOLL. Other modes may define differing
             ; requirements.
             ;
             pollwinner /
             ;
             )
```

Figure 20

Description This component provides a mechanism by which voters can

vote on provided choices. The outcome depends upon the POLL-MODE in effect.

The PARTICIPANT components in VPOLL requests provide information on each recipient who will be voting - both their identity through the CALENDAR-ADDRESS property and their votes through the VOTE components.

If specified, the "DTSTART" property defines the start or opening of the poll active period. If absent the poll is presumed to have started when created.

If "DTSTART" is present "DURATION" MAY be specified and indicates the duration, and hence the ending, of the poll. The value of the property MUST be a positive duration.

"DTEND" MAY be specified with or without "DTSTART" and indicates the ending of the poll. If DTEND is specified it MUST be later than the DTSTART or CREATED property.

If one or more VALARM components are included in the VPOLL they are not components to be voted on and MUST NOT contain a POLL-ITEM-ID property. VALARM sub-components may be used to provide warnings to the user when polls are due to start or end.

5.6.2. VOTE Component

Component name VOTE

Purpose This component provides a mechanism by which voters can vote on provided choices.

Conformance This component may be specified zero or more times in a PARTICIPANT component which identifies the voter.

Format Definition This property is defined by the following notation:

```
votec     = "BEGIN" ":" "VOTE" CRLF
            voteprop
            *eventc *todoc *journalc *freebusyc
            *availabilityc *alarmc *iana-comp *x-comp
            "END" ":" "VOTE" CRLF

voteprop = *(
           ;
           ; The following are REQUIRED,
           ; but MUST NOT occur more than once.
           ;
           pollitemid / response /
           ;
           ; The following are OPTIONAL,
           ; and MAY occur more than once.
           ;
           comment / x-prop / iana-prop
           ;
           )
```

Figure 21

Description This component appears inside the PARTICIPANT component with a PARTICIPANT-TYPE of VOTER to identify the voter. This component contains that participants responses.

The required and optional properties and their meanings will depend upon the POLL-MODE in effect.

For any POLL-MODE, POLL-ITEM-ID is used to associate the information to a choice supplied by the organizer. This means that each VOTE component only provides information about that choice.

If allowed by the POLL-MODE a VOTE component without a POLL-ITEM-ID may be provided in a REPLY to indicate a possible new choice or to provide information to the ORGANIZER - such as the respondees availability.

6. Poll Modes

The VPOLL component is intended to allow for various forms of polling. The particular form in efffect is indicated by the POLL-MODE property.

New poll modes can be registered by including a completed POLL-MODE Registration Template (see Section 10.3) in a published RFC.

6.1. POLL-MODE:BASIC

BASIC poll mode is the form of voting in which one possible outcome is chosen from a set of possibilities. Usually this will be represented as a number of possible event objects one of which will be selected.

6.1.1. Property restrictions

This poll mode has the following property requirements:

POLL-ITEM-ID Each contained sub-component that is being voted upon MUST contain a POLL-ITEM_ID property which is unique within the context of the POLL. The value MUST NOT be reused when events are removed and/or added to the poll.

POLL-WINNER On confirmation of the poll this property MUST be present and identifies the winning component.

6.1.2. Outcome reporting

To confirm the winner the POLL-WINNER property MUST be present and the STATUS MUST be set to CONFIRMED.

When the winning VEVENT or VTODO is not a scheduled entity, that is, it has no ORGANIZER or ATTENDEES it MUST be assigned an ORGANIZER property and a list of non-participating ATTENDEEs. This allows the winning entity to be distributed to the participants through iTip or some other protocol.

7. iTIP Extensions

This specification introduces a number of extensions to [RFC5546]. In group scheduling the parties involved are organizer and attendees. In VPOLL the parties are organizer and voters.

For many of the iTip processing rules the voters take the place of attendees.

7.1. Methods

There are some extensions to the behavior of iTip methods for a VPOLL object and two new methods are defined.

Method	Description
PUBLISH	No changes (yet)
REQUEST	Each child component MUST have a POLL-ITEM-ID property. Each set of components with the same POLL-ITEM-ID value represents one overall set of items to be voted on.
REPLY	There MUST be a single VPOLL component which MUST have: either one or more POLL-ITEM-ID properties with a RESPONSE param matching that from a REQUEST or a VFREEBUSY or VAVAILABILITY child component showing overall busy/available time. The VPOLL MUST have one voter only.
ADD	Not supported for VPOLL.
CANCEL	There MUST be a single VPOLL component with UID
	matching that of the poll being cancelled.
REFRESH	The organizer returns a METHOD:REQUEST with the current full state, or a METHOD:CANCEL or an error if no matching poll is found.
COUNTER	Not supported for VPOLL.
DECLINECOUNTER	Not supported for VPOLL.
POLLSTATUS	Used to send the current state of the poll to all voters. The VPOLL can contain a reduced set of properties but MUST contain DTSTAMP, SEQUENCE (if not 0), UID, ORGANIZER and PARTICIPANTS.

Table 1

The following table shows the above methods broken down by who can send them with VPOLL components.

Originator	Methods
Organizer	CANCEL, PUBLISH, REQUEST, POLLSTATUS
Voter	REPLY, REFRESH, REQUEST (only when delegating)

Table 2

7.2. Interoperability Models

Most of the standard iTip specification applies with respect to organizer and voters.

7.2.1. Delegation

TBD

7.2.2. Acting on Behalf of Other Calendar Users

TBD

7.2.3. Component Revisions

* Need to talk about what a change in SEQUENCE means
* Sequence change forces a revote.
* New voter - no sequence change
* Add another poll set or change poll item ids or any change to a child
* component - bump sequence

7.2.4. Message Sequencing

TBD

7.3. Application Protocol Elements

7.3.1. Methods for VPOLL Calendar Components

This section defines the property set restrictions for the method types that are applicable to the "VPOLL" calendar component. Each method is defined using a table that clarifies the property constraints that define the particular method.

The presence column uses the following values to assert whether a property is required or optional, and the number of times it may appear in the iCalendar object.

Presence Value	Description
1	One instance MUST be present.
1+	At least one instance MUST be present.
0	Instances of this property MUST NOT be present.
0+	Multiple instances MAY be present.
0 or 1	Up to 1 instance of this property MAY be present.

Table 3

The following summarizes the methods that are defined for the "VPOLL" calendar component.

Method	Description
PUBLISH	Post notification of an poll. Used primarily as a method of advertising the existence of a poll.
REQUEST	To make a request for a poll. This is an explicit invitation to one or more voters. Poll requests are also used to update, change or confirm an existing poll. Clients that cannot handle REQUEST MAY degrade the poll to view it as a PUBLISH. REQUEST SHOULD NOT be used just to set the status of the poll - POLLSTATUS provides a more compact approach.
REPLY	Reply to a poll request. Voters may set their RESPONSE parameter to supply the current vote in the range 0 to 100.
CANCEL	Cancel a poll.
REFRESH	A request is sent to an Organizer by a Voter asking for the latest version of a poll to be resent to the requester.
POLLSTATUS	Used to send the current state of the poll to all voters. The VPOLL can contain a reduced set of properties but MUST contain DTSTAMP, SEQUENCE (if not 0), UID, ORGANIZER and PARTICIPANT.

Table 4

7.3.2. Method: PUBLISH

The "PUBLISH" method in a "VPOLL" calendar component is an unsolicited posting of an iCalendar object. Any CU may add published components to their calendar. The "Organizer" MUST be present in a published iCalendar component. "Voters" MUST NOT be present. Its expected usage is for encapsulating an arbitrary poll as an iCalendar object. The "Organizer" may subsequently update (with another "PUBLISH" method) or cancel (with a "CANCEL" method) a previously published "VPOLL" calendar component.

Note Not clear how useful this is but needs some work on transmitting the current vote without any voter identification.

This method type is an iCalendar object that conforms to the following property constraints:

Component/ Property	Presence	Comment
METHOD	1	MUST equal PUBLISH.
VPOLL	1+	
DTSTAMP	1	
DTSTART	0 or 1	If present defines the start of the poll. Otherwise the poll starts when it is created and distributed.
ORGANIZER	1	
SUMMARY	1	Can be null.
UID	1	
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
ACCEPT-RESPONSE	0 or 1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0 or 1	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DTEND MUST NOT be

		present.
LAST-MODIFIED	0 or 1	
POLL-ITEM-ID	0	
POLL-MODE	0 or 1	
POLL-PROPERTIES	0 or 1	
PRIORITY	0 or 1	
RELATED-TO	0+	
RESOURCES	0+	
STATUS	0 or 1	MAY be one of COMPLETED/CONFIRMED/ CANCELLED.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
PARTICIPANT	0+	Only PARTICIPANT components with PARTICIPANT-TYPE not equal to "VOTER" - that is, no voters
REQUEST-STATUS	0	
VALARM	0+	
VEVENT	0+	Depending upon the poll mode in effect there MAY be candidate components included in the poll component.
VFREEBUSY	0	
VJOURNAL	0+	Depending upon the poll mode in effect there MAY be candidate components included in the poll component.
VTODO	0+	Depending upon the poll mode in effect there MAY be candidate components included in the poll

		component.
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	

Table 5: Constraints for a METHOD:PUBLISH of a VPOLL

7.3.3. Method: REQUEST

The "REQUEST" method in a "VPOLL" component provides the following scheduling functions:

* Invite "Voters" to respond to the poll.

* Change the items being voted upon.

* Complete or confirm the poll.

* Response to a "REFRESH" request.

* Update the details of an existing vpoll.

* Update the status of "Voters".

* Forward a "VPOLL" to another uninvited CU.

* For an existing "VPOLL" calendar component, delegate the role of "Voter" to another CU.

* For an existing "VPOLL" calendar component, change the role of "Organizer" to another CU.

The "Organizer" originates the "REQUEST". The recipients of the "REQUEST" method are the CUs voting in the poll, the "Voters". "Voters" use the "REPLY" method to convey votes to the "Organizer".

The "UID" and "SEQUENCE" properties are used to distinguish the various uses of the "REQUEST" method. If the "UID" property value in the "REQUEST" is not found on the recipient’s calendar, then the "REQUEST" is for a new "VPOLL" calendar component. If the "UID" property value is found on the recipient’s calendar, then the "REQUEST" is for an update, or a reconfirmation of the "VPOLL" calendar component.

For the "REQUEST" method only a single iCalendar object is permitted.

This method type is an iCalendar object that conforms to the following property constraints:

Component/ Property	Presence	Comment
METHOD	1	MUST be REQUEST.
VPOLL	1	
PARTICIPANT	1+	Identified as voters with the PARTICIPANT-TYPE=VOTER
DTSTAMP	1	
DTSTART	0 or 1	If present defines the start of the poll. Otherwise the poll starts when it is created and distributed.
ORGANIZER	1	
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
SUMMARY	1	Can be null.
UID	1	
ACCEPT-RESPONSE	0 or 1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.

DTEND	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DTEND MUST NOT be present.
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
POLL-ITEM-ID	0	
POLL-MODE	0 or 1	
POLL-PROPERTIES	0 or 1	
PRIORITY	0 or 1	
RELATED-TO	0+	
REQUEST-STATUS	0	
RESOURCES	0+	
STATUS	0 or 1	MAY be one of COMPLETED/CONFIRMED/ CANCELLED.
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0+	Depending upon the poll mode in

		effect there MAY be candidate components included in the poll component.
VFREEBUSY	0	
VJOURNAL	0+	Depending upon the poll mode in effect there MAY be candidate components included in the poll component.
VTODO	0+	Depending upon the poll mode in effect there MAY be candidate components included in the poll component.

Table 6: Constraints for a METHOD:REQUEST of a VPOLL

7.3.3.1. Rescheduling a poll

The "REQUEST" method may be used to reschedule a poll, that is force a revote. A rescheduled poll involves a change to the existing poll in terms of its time the components being voted on may have changed. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar but that the "SEQUENCE" (or "DTSTAMP") property value in the "REQUEST" method is greater than the value for the existing poll, then the "REQUEST" method describes a rescheduling of the poll.

7.3.3.2. Updating or Reconfirmation of a Poll

The "REQUEST" method may be used to update or reconfirm a poll. An update to an existing poll does not involve changes to the time or candidates, and might not involve a change to the location or description for the poll. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar and that the "SEQUENCE" property value in the "REQUEST" is the same as the value for the existing poll, then the "REQUEST" method

describes an update of the poll details, but not a rescheduling of the POLL.

The update "REQUEST" method is the appropriate response to a "REFRESH" method sent from a "Voter" to the "Organizer" of a poll.

The "Organizer" of a poll may also send unsolicited "REQUEST" methods. The unsolicited "REQUEST" methods may be used to update the details of the poll without rescheduling it, to update the "RESPONSE" parameter of "Voters", or to reconfirm the poll.

7.3.3.3. Confirmation of a Poll

The "REQUEST" method may be used to confirm a poll, that is announce the winner in BASIC mode. The STATUS MUST be set to CONFIRMED and for BASIC mode a VPOLL POLL-WINNER property must be provided with the poll-id of the winning component.

7.3.3.4. Closing a Poll

The "REQUEST" method may be used to close a poll, that is indicate voting is completed. The STATUS MUST be set to COMPLETED.

7.3.3.5. Delegating a Poll to Another CU

Some calendar and scheduling systems allow "Voters" to delegate the vote to another "Calendar User". iTIP supports this concept using the following workflow. Any "Voter" may delegate their right to vote in a poll to another CU. The implication is that the delegate participates in lieu of the original "Voter", NOT in addition to the "Voter". The delegator MUST notify the "Organizer" of this action using the steps outlined below. Implementations may support or restrict delegation as they see fit. For instance, some implementations may restrict a delegate from delegating a "REQUEST" to another CU.

The "Delegator" of a poll forwards the existing "REQUEST" to the "Delegate". The "REQUEST" method MUST include a "Voter" property with the calendar address of the "Delegate". The "Delegator" MUST also send a "REPLY" method to the "Organizer" with the "Delegator's" "Voter" property "DELEGATED-TO" parameter set to the calendar address of the "Delegate". Also, a new "Voter" property for the "Delegate" MUST be included and must specify the calendar user address set in the "DELEGATED-TO" parameter, as above.

In response to the request, the "Delegate" MUST send a "REPLY" method to the "Organizer", and optionally to the "Delegator". The "REPLY"

method SHOULD include the "Voter" property with the "DELEGATED-FROM" parameter value of the "Delegator's" calendar address.

The "Delegator" may continue to receive updates to the poll even though they will not be attending. This is accomplished by the "Delegator" setting their "role" attribute to "NON-PARTICIPANT" in the "REPLY" to the "Organizer".

7.3.3.6. Changing the Organizer

The situation may arise where the "Organizer" of a "VPOLL" is no longer able to perform the "Organizer" role and abdicates without passing on the "Organizer" role to someone else. When this occurs, the "Voters" of the "VPOLL" may use out-of-band mechanisms to communicate the situation and agree upon a new "Organizer". The new "Organizer" should then send out a new "REQUEST" with a modified version of the "VPOLL" in which the "SEQUENCE" number has been incremented and the "ORGANIZER" property has been changed to the new "Organizer".

7.3.3.7. Sending on Behalf of the Organizer

There are a number of scenarios that support the need for a "Calendar User" to act on behalf of the "Organizer" without explicit role changing. This might be the case if the CU designated as "Organizer" is sick or unable to perform duties associated with that function. In these cases, iTIP supports the notion of one CU acting on behalf of another. Using the "SENT-BY" parameter, a "Calendar User" could send an updated "VPOLL" "REQUEST". In the case where one CU sends on behalf of another CU, the "Voter" responses are still directed back towards the CU designated as "Organizer".

7.3.3.8. Forwarding to an Uninvited CU

A "Voter" invited to a "VPOLL" calendar component may send the "VPOLL" calendar component to another new CU not previously associated with the "VPOLL" calendar component. The current "Voter" participating in the "VPOLL" calendar component does this by forwarding the original "REQUEST" method to the new CU. The new CU can send a "REPLY" to the "Organizer" of the "VPOLL" calendar component. The reply contains a "Voter" property for the new CU.

The "Organizer" ultimately decides whether or not the new CU becomes part of the poll and is not obligated to do anything with a "REPLY" from a new (uninvited) CU. If the "Organizer" does not want the new CU to be part of the poll, the new "Voter" property is not added to the "VPOLL" calendar component. The "Organizer" MAY send the CU a "CANCEL" message to indicate that they will not be added to the poll.

If the "Organizer" decides to add the new CU, the new "Voter" property is added to the "VPOLL" calendar component. Furthermore, the "Organizer" is free to change any "Voter" property parameter from the values supplied by the new CU to something the "Organizer" considers appropriate. The "Organizer" SHOULD send the new CU a "REQUEST" message to inform them that they have been added.

When forwarding a "REQUEST" to another CU, the forwarding "Voter" MUST NOT make changes to the original message.

7.3.3.9. Updating Voter Status

The "Organizer" of an poll may also request updated status from one or more "Voters". The "Organizer" sends a "REQUEST" method to the "Voter" and sets the "RSVP=TRUE" property parameter on the PARTICIPANT CALENDAR-ADDRESS. The "SEQUENCE" property for the poll is not changed from its previous value. A recipient will determine that the only change in the "REQUEST" is that their "RSVP" property parameter indicates a request for updated status. The recipient SHOULD respond with a "REPLY" method indicating their current vote with respect to the "REQUEST".

7.3.4. Method: REPLY

The "REPLY" method in a "VPOLL" calendar component is used to respond (e.g., accept or decline) to a "REQUEST" or to reply to a delegation "REQUEST". When used to provide a delegation response, the "Delegator" SHOULD include the calendar address of the "Delegate" on the "DELEGATED-TO" property parameter of the "Delegator's" "CALENDAR-ADDRESS" property. The "Delegate" SHOULD include the calendar address of the "Delegator" on the "DELEGATED-FROM" property parameter of the "Delegate's" "CALENDAR-ADDRESS" property.

The "REPLY" method is also used when processing of a "REQUEST" fails. Depending on the value of the "REQUEST-STATUS" property, no action may have been performed.

The "Organizer" of a poll may receive the "REPLY" method from a CU not in the original "REQUEST". For example, a "REPLY" may be received from a "Delegate" to a poll. In addition, the "REPLY" method may be received from an unknown CU (a "Party Crasher"). This uninvited "Voter" may be accepted, or the "Organizer" may cancel the poll for the uninvited "Voter" by sending a "CANCEL" method to the uninvited "Voter".

A "Voter" MAY include a message to the "Organizer" using the "COMMENT" property. For example, if the user indicates a low interest and wants to let the "Organizer" know why, the reason can be expressed in the "COMMENT" property value.

The "Organizer" may also receive a "REPLY" from one CU on behalf of another. Like the scenario enumerated above for the "Organizer", "Voters" may have another CU respond on their behalf. This is done using the "SENT-BY" parameter.

The optional properties listed in the table below (those listed as "0+" or "0 or 1") MUST NOT be changed from those of the original request. (But see comments on VFREEBUSY and VAVAILABILITY)

This method type is an iCalendar object that conforms to the following property constraints:

Component/ Property	Presence	Comment
METHOD	1	MUST be REPLY.
VPOLL	1+	All components MUST have the same UID.
PARTICIPANT	1	Identifies the Voter replying.
DTSTAMP	1	
ORGANIZER	1	
UID	1	MUST be the UID of the original REQUEST.
SEQUENCE	0 or 1	If non-zero, MUST be the sequence number of the original REQUEST. MAY be present if 0.
ACCEPT-RESPONSE	0 or 1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	

COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DTSTART	0 or 1	
DURATION	0 or 1	If present, DTEND MUST NOT be present.
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
POLL-ITEM-ID	1+	One per item being voted on.
POLL-MODE	0	
POLL-PROPERTIES	0	
PRIORITY	0 or 1	
RELATED-TO	0+	
RESOURCES	0+	
REQUEST-STATUS	0+	
STATUS	0 or 1	
SUMMARY	0 or 1	
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	

X-PROPERTY	0+	
VALARM	0	
VTIMEZONE	0 or 1	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0 or 1	A voter may respond with a VFREEBUSY component indicating that the ORGANIZER may select some other time which is not marked as busy.
VAVAILABILITY	0	A voter may respond with a VAVAILABILITY component indicating that the ORGANIZER may select some other time which is shown as available.
VJOURNAL	0	
VTODO	0	

Table 7: Constraints for a METHOD:REPLY of a VPOLL

7.3.5. Method: CANCEL

The "CANCEL" method in a "VPOLL" calendar component is used to send a cancellation notice of an existing poll request to the affected "Voters". The message is sent by the "Organizer" of the poll.

The "Organizer" MUST send a "CANCEL" message to each "Voter" affected by the cancellation. This can be done using a single "CANCEL" message for all "Voters" or by using multiple messages with different subsets of the affected "Voters" in each.

When a "VPOLL" is cancelled, the "SEQUENCE" property value MUST be incremented as described in Section 7.2.3.

Once a CANCEL message has been sent to all voters no further voting may take place. The poll is considered closed.

This method type is an iCalendar object that conforms to the following property constraints:

Component/ Property	Presence	Comment
METHOD	1	MUST be CANCEL.
VPOLL	1+	All must have the same UID.
PARTICIPANT	0+	MUST include some or all Voters being removed from the poll. MUST include some or all Voters if the entire poll is cancelled.
UID	1	MUST be the UID of the original REQUEST.
DTSTAMP	1	
ORGANIZER	1	
SEQUENCE	1	
ATTACH	0+	
ACCEPT-RESPONSE	0	
COMMENT	0+	
COMPLETED	0 or 1	
CATEGORIES	0+	
CLASS	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DTSTART	0 or 1	

DURATION	0 or 1	If present, DTEND MUST NOT be present.
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
POLL-ITEM-ID	0	
POLL-MODE	0	
POLL-PROPERTIES	0	
PRIORITY	0 or 1	
RELATED-TO	0+	
RESOURCES	0+	
STATUS	0 or 1	MUST be set to CANCELLED to cancel the entire event. If uninviting specific Attendees, then MUST NOT be included.
SUMMARY	0 or 1	
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
REQUEST-STATUS	0	
VALARM	0	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VTODO	0	

VJOURNAL	0	
VEVENT	0	
VFREEBUSY	0	

Table 8: Constraints for a METHOD:CANCEL of a VPOLL

7.3.6. Method: REFRESH

The "REFRESH" method in a "VPOLL" calendar component is used by "Voters" of an existing event to request an updated description from the poll "Organizer". The "REFRESH" method must specify the "UID" property of the poll to update. The "Organizer" responds with the latest description and version of the poll.

This method type is an iCalendar object that conforms to the following property constraints:

Component/Property	Presence	Comment
METHOD	1	MUST be REFRESH.
VPOLL	1	
PARTICIPANT	1	MUST identify the requester as a voter.
DTSTAMP	1	
ORGANIZER	1	
UID	1	MUST be the UID associated with original REQUEST.
COMMENT	0+	
COMPLETED	0	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
ACCEPT-RESPONSE	0	

ATTACH	0	
CATEGORIES	0	
CLASS	0	
CONTACT	0	
CREATED	0	
DESCRIPTION	0	
DTEND	0	
DTSTART	0	
DURATION	0	
GEO	0	
LAST-MODIFIED	0	
LOCATION	0	
POLL-ITEM-ID	0	
POLL-MODE	0	
POLL-PROPERTIES	0	
PRIORITY	0	
RELATED-TO	0	
REQUEST-STATUS	0	
RESOURCES	0	
SEQUENCE	0	
STATUS	0	
SUMMARY	0	
URL	0	
VALARM	0	

VTIMEZONE	0+	
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VTODO	0	
VJOURNAL	0	
VEVENT	0	
VFREEBUSY	0	

Table 9: Constraints for a METHOD:REFRESH of a VPOLL

7.3.7. Method: POLLSTATUS

The "POLLSTATUS" method in a "VPOLL" calendar component is used to inform recipients of the current status of the poll in a compact manner. The "Organizer" MUST be present in the confirmed poll component. All "Voters" MUST be present. The selected component(s) according to the poll mode SHOULD NOT be present in the poll component. Clients receiving this message may store the confirmed items in their calendars.

This method type is an iCalendar object that conforms to the following property constraints:

Component/ Property	Presence	Comment
METHOD	1	MUST equal POLLSTATUS.
VPOLL	1+	
PARTICIPANT	1+	The voters containing their current vote
COMPLETED	0 or 1	Only present for a completed poll
DTSTAMP	1	
DTSTART	0 or 1	
ORGANIZER	1	

SUMMARY	1	Can be null.
UID	1	
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
ACCEPT-RESPONSE	0	
ATTACH	0	
CATEGORIES	0	
CLASS	0	
COMMENT	0+	
CONTACT	0	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DTEND MUST NOT be present.
LAST-MODIFIED	0 or 1	
POLL-ITEM-ID	0	
POLL-MODE	0 or 1	
POLL-PROPERTIES	0	
PRIORITY	0 or 1	
RELATED-TO	0+	
RESOURCES	0+	
STATUS	0 or 1	MAY be one of TENTATIVE/CONFIRMED/ CANCELLED.
URL	0 or 1	

IANA-PROPERTY	0+	
X-PROPERTY	0+	
REQUEST-STATUS	0	
VALARM	0+	
VEVENT	0	All candidate components SHOULD NOT be present.
VFREEBUSY	0	
VJOURNAL	0	All candidate components SHOULD NOT be present.
VTODO	0	All candidate components SHOULD NOT be present.
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	

Table 10: Constraints for a METHOD:POLLSTATUS of a VPOLL

8. CalDAV Extensions

This specification extends [RFC4791] in that it defines a new component and new iCalendar properties to be supported and requires extra definitions related to time-ranges and reports.

Additionally, it extends [RFC6638] as it a VPOLL component is a schedulable entity.

8.1. Calendar Collection Properties

This section defines new CalDAV properties for calendar collections.

8.1.1. CALDAV:supported-vpoll-component-sets

Name supported-vpoll-component-sets

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

Purpose Specifies the calendar component types (e.g., VEVENT, VTODO, etc.) and combination of types that may be included in a VPOLL component.

Conformance This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in [RFC2518]).

Description The CALDAV:supported-vpoll-component-sets property is used to specify restrictions on the calendar component types that VPOLL components may contain in a calendar collection.

It also specifies the combination of allowed component types.

Any attempt by the client to store VPOLL components with component types or combinations of types not listed in this property, if it exists, MUST result in an error, with the "CALDAV:supported-vpoll-component-sets" precondition Section 8.2 being violated. Since this property is protected, it cannot be changed by clients using a PROPPATCH request. However, clients can initialize the value of this property when creating a new calendar collection with MKCALENDAR. In the absence of this property, the server MUST accept all component types, and the client can assume that all component types are accepted.

Definition

```
<!ELEMENT supported-vpoll-component-sets
     (supported-vpoll-component-set*) >

<!ELEMENT supported-vpoll-component-set (comp+)>
```

Figure 22

```
<C:supported-vpoll-component-sets
     xmlns:C="urn:ietf:params:xml:ns:caldav">

  <!-- VPOLLs with VEVENT, VFREEBUSY or VTODO -->
  <C:supported-vpoll-component-set>
    <C:comp name="VEVENT" />
    <C:comp name="VFREEBUSY" />
    <C:comp name="VTODO" />
  </C:supported-vpoll-component-set>

  <!-- VPOLLs with just VEVENT or VFREEBUSY -->
  <C:supported-vpoll-component-set>
    <C:comp name="VEVENT" />
    <C:comp name="VFREEBUSY" />
  </C:supported-vpoll-component-set>

  <!-- VPOLLs with just VEVENT -->
  <C:supported-vpoll-component-set>
    <C:comp name="VEVENT" />
  </C:supported-vpoll-component-set>

  <!-- VPOLLs with just VTODO -->
  <C:supported-vpoll-component-set>
    <C:comp name="VTODO" />
  </C:supported-vpoll-component-set>
</C:supported-vpoll-component-sets>
```

Figure 23

8.1.2. CALDAV:vpoll-max-items

Name vpoll-max-items

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

Purpose Provides a numeric value indicating the maximum number of items that may be contained in any instance of a VPOLL calendar object resource stored in the calendar collection.

Conformance This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in [RFC2518]).

Description The CALDAV:vpoll-max-items is used to specify a numeric value that indicates the maximum number of iCalendar components in any one instance of a VPOLL calendar object resource stored in a calendar collection. Any attempt to store a calendar object resource with more components per instance than this value MUST

result in an error, with the CALDAV: vpoll-max-items precondition Section 8.2 being violated. In the absence of this property, the client can assume that the server can handle any number of items in a VPOLL calendar component.

Definition

```
<!ELEMENT vpoll-max-items (#PCDATA)>
PCDATA value: a numeric value (integer greater than zero)
```

Figure 24

```
<C:vpoll-max-items xmlns:C="urn:ietf:params:xml:ns:caldav"
>25</C:vpoll-max-items>
```

Figure 25

8.1.3. CALDAV:vpoll-max-active

Name vpoll-max-active

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

Purpose Provides a numeric value indicating the maximum number of active vpolls at any one time.

Conformance This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in [RFC2518]).

Description The CALDAV:vpoll-max-active is used to specify a numeric value that indicates the maximum number of active VPOLLs at any one time. Any attempt to store a new active VPOLL calendar object resource which results in exceeding this limit MUST result in an error, with the "CALDAV:vpoll-max-active" precondition Section 8.2 being violated. In the absence of this property, the client can assume that the server can handle any number of active VPOLLs.

Definition

```
<!ELEMENT vpoll-max-active (#PCDATA)>
PCDATA value: a numeric value (integer greater than zero)
```

Figure 26

```
<C:vpoll-max-active xmlns:C="urn:ietf:params:xml:ns:caldav"
>25</C:vpoll-max-active>
```

Figure 27

8.1.4. CALDAV:vpoll-max-voters

Name "vpoll-max-voters"

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

Purpose Provides a numeric value indicating the maximum number of voters for any instance of a VPOLL calendar object resource stored in the calendar collection.

Conformance This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND "DAV:allprop" request (as defined in [RFC2518]).

Description The "CALDAV:vpoll-max-voters" is used to specify a numeric value that indicates the maximum number of voters for any one instance of a VPOLL calendar object resource stored in a calendar collection. Any attempt to store a calendar object resource with more voters per instance than this value MUST result in an error, with the CALDAV: "vpoll-max-voters" precondition Section 8.2 being violated. In the absence of this property, the client can assume that the server can handle any number of voters in a VPOLL calendar component.

Definition

```
<!ELEMENT vpoll-max-voters (#PCDATA)>
PCDATA value: a numeric value (integer greater than zero)
```

Figure 28

```
<C:vpoll-max-voters xmlns:C="urn:ietf:params:xml:ns:caldav"
>25</C:vpoll-max-voters>
```

Figure 29

8.1.5. CalDAV:even-more-properties

8.1.6. Extensions to CalDAV scheduling

This specification extends [RFC6638].

Each section of Appendix A "Scheduling Privileges Summary" is extended to include VPOLL.

Any reference to the ATTENDEE property should be read to include the CALENDAR-ADDRESS property contained in the PARTICIPANT compoents. That is, for scheduling purposes the CALENDAR-ADDRESS property is handled in exactly the same manner as the ATTENDEE property.

8.2. Additional Preconditions for PUT, COPY, and MOVE

This specification creates additional Preconditions for PUT, COPY, and MOVE methods. These preconditions apply when a PUT operation of a VPOLL calendar object resource into a calendar collection occurs, or when a COPY or MOVE operation of a calendar object resource into a calendar collection occurs, or when a COPY or MOVE operation occurs on a calendar collection.

The new preconditions are:

(CALDAV:supported-vpoll-component-sets) The VPOLL resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST contain a type or combination of calendar component that is supported in the targeted calendar collection;

(CALDAV:vpoll-max-items) The VPOLL resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have a number of sub-components (excluding VTIMEZONE) less than or equal to the value of the "CALDAV:vpoll-max-items" property value Section 8.1.2 on the calendar collection where the resource will be stored;

(CALDAV:vpoll-max-active) The PUT request, or COPY or MOVE request, MUST not result in the number of active VPOLLs being greater than the value of the "CALDAV:vpoll-max-active" property value Section 8.1.3 on the calendar collection where the resource will be stored;

(CALDAV:vpoll-max-voters) The VPOLL resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have a number of voters represented by PARTICIPANT components less than or equal to the value of the "CALDAV:vpoll-max-voters" property value Section 8.1.4 on the calendar collection where the resource will be stored;

8.3. CalDAV:calendar-query Report

This allows the retrieval of VPOLLs and their included components. The query specification allows queries to be directed at the contained sub-components. For VPOLL queries this feature is disallowed. Time-range queries can only target the vpoll component itself.

8.3.1. Example: Partial Retrieval of VPOLL

In this example, the client requests the server to return specific components and properties of the VPOLL components that overlap the time range from December 4, 2012, at 00:00:00 A.M. UTC to December 5, 2012, at 00:00:00 A.M. UTC. In addition, the "DAV:getetag" property is also requested and returned as part of the response. Note that due to the CALDAV: calendar-data element restrictions, the DTSTAMP property in VPOLL components has not been returned, and the only property returned in the VCALENDAR object is VERSION.

>> Request <<

```
REPORT /cyrus/work/ HTTP/1.1
Host: cal.example.com
Depth: 1
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:calendar-query xmlns:D="DAV:"
                  xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:prop>
    <D:getetag/>
    <C:calendar-data>
      <C:comp name="VCALENDAR">
        <C:prop name="VERSION"/>
        <C:comp name="VPOLL">
          <C:prop name="SUMMARY"/>
          <C:prop name="UID"/>
          <C:prop name="DTSTART"/>
          <C:prop name="DTEND"/>
          <C:prop name="DURATION"/>
        </C:comp>

      </C:comp>
    </C:calendar-data>
  </D:prop>
  <C:filter>
    <C:comp-filter name="VCALENDAR">
      <C:comp-filter name="VPOLL">
        <C:time-range start="20121204T000000Z"
                      end="20121205T000000Z"/>
      </C:comp-filter>
    </C:comp-filter>
  </C:filter>
</C:calendar-query>
```

```
   >> Response <<

   HTTP/1.1 207 Multi-Status
   Date: Sat, 11 Nov 2012 09:32:12 GMT
   Content-Type: application/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:"
              xmlns:C="urn:ietf:params:xml:ns:caldav">
     <D:response>
       <D:href>http://cal.example.com/cyrus/work/poll2.ics</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"fffff-abcd2"</D:getetag>
           <C:calendar-data>BEGIN:VCALENDAR
   VERSION:2.0
   BEGIN:VPOLL
   DTSTART;TZID=US/Eastern:20121202T120000
   DURATION:PT4D
   SUMMARY:Poll #2
   UID:00959BC664CA650E933C892C@example.com
   END:VPOLL
   END:VCALENDAR
   </C:calendar-data>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
     </D:response>
     <D:response>
       <D:href>http://cal.example.com/cyrus/work/poll3.ics</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"fffff-abcd3"</D:getetag>
           <C:calendar-data>BEGIN:VCALENDAR

   VERSION:2.0
   PRODID:-//Example Corp.//CalDAV Client//EN
   BEGIN:VPOLL
   DTSTART;TZID=US/Eastern:20121204T100000
   DURATION:PT4D
   SUMMARY:Poll #3
   UID:DC6C50A017428C5216A2F1CD@example.com
   END:VPOLL
   END:VCALENDAR
   </C:calendar-data>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
```

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

Figure 30

8.4. CalDAV time ranges

"CALDAV:time-range XML Element" in [RFC4791] describes how to specify time ranges to limit the set of calendar components returned by the server. This specification extends [RFC4791] to describe the meaning of time ranges for VPOLL

A VPOLL component is said to overlap a given time range if the condition for the corresponding component state specified in the table below is satisfied. The conditions depend on the presence of the DTSTART, DURATION, DTEND, COMPLETED and CREATED properties in the VPOLL component. Note that, as specified above, the DTEND value MUST be a DATE-TIME value equal to or after the DTSTART value if specified.

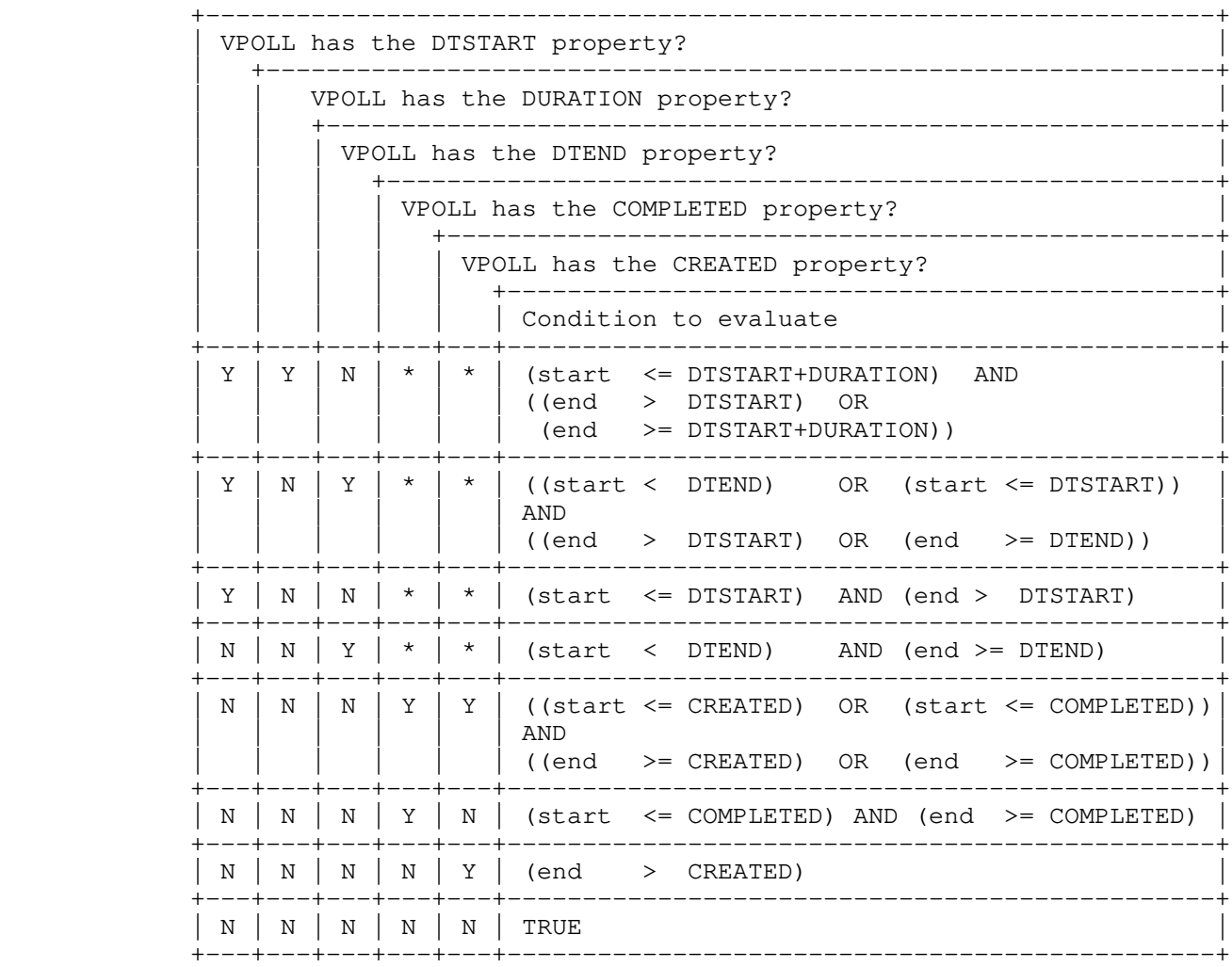

Figure 31

9. Security Considerations

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

10. IANA Considerations

10.1. Parameter Registrations

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

Property Parameter	Status	Reference
REQUIRED	Current	Section 5.4.1
STAY-INFORMED	Current	Section 5.4.2

Table 11

10.2. Property Registrations

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

Property	Status	Reference
ACCEPT-RESPONSE	Current	Section 5.5.7
POLL-ITEM-ID	Current	Section 5.5.3
POLL-MODE	Current	Section 5.5.4
POLL-PROPERTIES	Current	Section 5.5.5
POLL-WINNER	Current	Section 5.5.6
RESPONSE	Current	Section 5.5.8

Table 12

10.3. POLL-MODE Registration Template

A poll mode is defined by completing the following template.

Poll mode name The name of the poll mode.

Purpose The purpose of the poll mode. Give a short but clear description.

Reference A reference to the RFC in which the poll mode is defined

10.4. POLL-MODE Registrations

This document defines the following registered poll modes.

Poll mode name	Purpose	Reference
BASIC	To provide simple voting for a single outcome from a number of candidates.	Current

Table 13

11. Normative references

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

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

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", IETF RFC 4791, IETF 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)", IETF RFC 5545, IETF 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)", IETF RFC 5546, IETF RFC 5546, DOI 10.17487/RFC5546, December 2009, <https://www.rfc-editor.org/info/rfc5546>.

[RFC6047] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", IETF RFC 6047, IETF RFC 6047, DOI 10.17487/RFC6047, December 2010, <https://www.rfc-editor.org/info/rfc6047>.

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

[I-D.ietf-calext-eventpub-extensions] Douglass, M., "Event Publishing Extensions to iCalendar", IETF I-D.ietf-calext-eventpub-extensions, IETF I-D.ietf-calext-eventpub-extensions, October 2019.

Appendix A. Open issues

public-comment: Not documented and was a parameter on something. Really sounds like a PARTICIPANT or VOTE property

Notifications: Need to do a section on what Notifications to support. A. VPOLL is about to end and you haven't voted on it yet. Instead reuse VALARMS to notify the user?

Future: Restarting a confirmed/completed VPOLL What to do with changes to STATUS:CONFIRMED? Allow them or not? What do to that poll had a winning event or todo. Stress VPOLL UID MUST be unique Changing status back from CONFIRMED MUST adjust status of any events booked as a result of confirmation. MUST winning event be cancelled for POLL-MODE basic? No - voter has indicated now unable to attend - want to revote

Future: Voting on a confirmed/completed VPOLL Can a voter vote after completion? May be unable to attend and wants to indicate. Requires retention of VPOLL retention period Removed status

ORGANIZER/ATTENDEE validity Can a user create a poll with scheduled events where that user's isn't the organizer of the poll? So is there a requirement that the account that poll is on is able to create each one of the resources in the poll? i.e. I can't create a poll with a set of events where I am just the attendee of the events. Are there any other restrictions for components in a VPOLL? Add to security consideration

Update to existing event after poll confirm When voting on existing event - winning properties ONLY are merged in to the real event.

Need to write down what isn't valid in a VPOLL a. Can't change POLL-MODE

Guide for ATTENDEE roles chair, NON-PARTICIPANT etc

? - some iTip notes On confirm - send itip if appropriate (PUBLISH) - all non-participating - shared - feeds Organizer can specify where result is? Confirm can specify that itip is sent - ITIP / NONE - parameter ? on POLL-WINNER

Need to add example of freebusy in response

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//BedeworkCaldavTest//BedeworkCaldavTest
METHOD: REPLY
BEGIN:VPOLL
ORGANIZER:mailto:douglm@mysite.edu
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:eric@example.com
UID:sched01-1234567890
DTSTAMP:20120101T010000Z
SEQUENCE:0
SUMMARY:What to do this week
BEGIN:VFREEBUSY
.......
END:VFREEBUSY
END:PARTICIPANT
END:VPOLL
END:VCALENDAR
```

Figure 32

Appendix B. Change log

Calext V01: 2019-10-17 MD Replace VVOTER and VOTER with PARTICIPANT.

Calext V00: 2019-05-17 MD First calext version. Moved source to metanorma. No changes to specification.

V03: 2014-10-28 MD

* Add VVOTER and VOTE components.
* Add RESPONSE property.
* Remove RESPONSE parameter from VOTER.

V03: 2014-05-12 MD

* Add reply-url property and required parameter.
* Fix ACCEPT-RESPONSE definition.

V02: 2014-05-12 MD

* Typos fixed, clarifications made.

* Removed spurious COMMENT param. Switched some to PUBLIC-COMMENT

* Changed STAY-INFORMED to remove boolean value type and state explicit TRUE/FALSE values.

* iTip: Allow VPOLL DTSTART to be optional and allow VAVAILABILITY as subcomponent

* iTip: fix broken table cells

* Add POLL-PROPERTIES, POLL-WINNER to 5545 extensions table

* Added Caldav scheduling section

V01: 2013-08-07 MD

* Removed method CONFIRM

* Removed pollitemid from VPOLL abnf. Added text for pollwinner

* Added POLL-WINNER and verbiage

* Added STATUS values

* Added RELTYPE=POLL

* Added supported-vpoll-component-sets

* Added CalDAV related parameters to VOTER

* Removed bad CalDAV query example. State that queries cannot target the sub-components.

Initial version: 2012-11-02 MD

Authors' Addresses

Eric York

Email: eric.york@gmail.com

Cyrus Daboo

Email: cyrus@daboo.name

Michael Douglass

Email: mikeadouglass@gmail.com