Calendaring extensions
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2018

N. Jenkins
R. Stepanek
FastMail

March 2, 2018

JSCalendar: A JSON representation of calendar data
draft-ietf-calext-jscalendar-01

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 3, 2018.

Table of Contents

1. Introduction

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

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

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

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

1.1. Relation to the iCalendar format

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

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

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

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

1.2. Notational Conventions

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

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

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

2. JSCalendar objects

This section describes the calendar object types specified by JSCalendar.

2.1. JSEvent

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

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

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

2.2. JSTask

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

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

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

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

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

2.3. JSGroup

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

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

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

3. Structure of JSCalendar objects

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

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

3.1. Type signatures

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

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

3.2. Data Types

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

3.2.1. UTCDate

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

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

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

3.2.2. LocalDate

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

3.2.3. Duration

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

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

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

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

3.2.4. PatchObject

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

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

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

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

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

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

The value associated with each pointer is either:

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

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

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

3.2.5. Normalisation and equivalence

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

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

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

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

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

3.3. Custom property extensions and values

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

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

4. Common JSCalendar properties

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

4.1. Metadata properties

4.1.1. @type

Type: "String"

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

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

A valid JSCalendar object MUST include this property.

4.1.2. uid

Type: "String"

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

A valid JSCalendar object MUST include this property.

4.1.3. relatedTo

Type: "String[Relation]|null"

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

A *Relation* object has the following properties:

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

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

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

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

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

  * "parent": This object is part of the overall linked object.

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

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

4.1.4. prodId

Type: "String|null"

The identifier for the product that created the JSCalendar object.

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

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

4.1.5. created

Type: "UTCDate|null"

The date and time this object was initially created.

4.1.6. updated

Type: "UTCDate"

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

4.1.7. sequence

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

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

4.1.8. method

Type: "String|null"

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

4.2. What and where properties

4.2.1. title

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

A short summary of the object.

4.2.2. description

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

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

4.2.3. htmlDescription

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

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

4.2.4. locations

Type: "String[Location]|null"

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

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

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

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

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

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

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

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

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

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

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

* "audio": audio conferencing

* "chat": chat or instant messaging

* "screen": screen sharing

* "video": video conferencing

* any vendor-prefixed custom value

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

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

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

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

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

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

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

4.2.5. links

Type: "String[Link]|null"

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

A *Link* object has the following properties:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2.6. locale

Type: "String|null"

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

4.2.7. keywords

Type: "String[]|null"

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

4.2.8. categories

Type: "String[]|null"

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

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

4.2.9. color

Type: "String|null"

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

4.3. Recurrence properties

4.3.1. recurrenceRule

Type: "Recurrence"

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

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

A Recurrence object has the following properties:

o *frequency*: "String" This MUST be one of the following values:

  * "yearly"

  * "monthly"

  * "weekly"

  * "daily"

  * "hourly"

  * "minutely"

  * "secondly"

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.3.2. recurrenceOverrides

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

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

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

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

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

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

4.4. Sharing and scheduling properties

4.4.1. priority

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

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

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

4.4.2. freeBusyStatus

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

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

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

4.4.3. privacy

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

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

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

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

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

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

  * title

  * description

  * htmlDescription

  * locations

  * links

  * locale

  * localizations

  * participants

  * replyTo

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

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

4.4.4. replyTo

Type: "String[String]|null"

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

The following methods are defined:

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

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

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

Table 1: replyTo URI Template variables

4.4.5. participants

Type: "String[Participant]|null"

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

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

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

- *email*: "String" The email address for the participant.

- *kind*: "String"(optional, defaults to "unknown") What kind of entity this participant is.

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

  * "individual": a single person

  * "group": a collection of people invited as a whole

  * "resource": a non-human resource, e.g. a projector

  * "location": a physical location involved in the calendar object that needs to be scheduled, e.g. a conference room.

  * "unknown": no information is available about the kind of this participant.

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

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

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

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

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

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

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

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

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

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

  * "accepted": The invited participant will participate.

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

  * "tentative": The invited participant may participate.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.5. Alerts properties

4.5.1. useDefaultAlerts

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

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

4.5.2. alerts

Type: "String[Alert]|null"

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

An *Alert* Object has the following properties:

- *relativeTo*: "String" (optional, defaults to "before-start") Specifies where the offset is relative to for the alarm to trigger. The value MUST be one of:

  - "before-start"

  - "after-start"

  - "before-end"

  - "after-end"

- *offset*: "Duration" The offset from the start and end/due of the calendar object to fire the alert. If the calendar object does not define a time-zone, the user's default time-zone SHOULD be used when determining the offset, if known. Otherwise, the time-zone to use is implementation specific.

- *action*: "DisplayAction|EmailAction|UnknownAction"

  Describes how to alert the user.

  A *DisplayAction* means a message (which is service dependent, but SHOULD include the title and start or due time of the calendar object) SHOULD be shown to the user on any client connected to this account at the specified time. How this message is formatted (and any sound or other method of drawing the user's attention) is client specific. It has the following properties:

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

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

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

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

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

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

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

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

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

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

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

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

  An *Emailer* object has the following properties:

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

  + email: String The email address of the recipient.

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

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

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

* *attachments*: "String[]|null" (optional) A list of link identifiers in the JSCalendar object links property. Included attachments SHOULD be embedded in the MIME message with the "Content-Disposition" header value set to "attachment" (see [RFC2183]). Implementations MAY refuse to include one or more attachments when building an alert email, in which case they

MUST ignore the contents of the *attachments* property (e.g. they MUST NOT include a subset of attachments).

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

4.6. Multilingual properties

4.6.1. localizations

Type: "String[PatchObject]|null"

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

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

- title
- description
- htmlDescription
- keywords
- Location name
- Location description
- Link title
- EmailAction alert subject
- EmailAction alert textBody
- EmailAction alert htmlBody
- any Link properties entry except the "display" field
- any UnknownAction alert property except the "type" field

- any unknown or vendor-specific property (if not defined otherwise in a future RFC or vendor-specific extension).

5. Type-specific JSCalendar properties

5.1. JSEvent properties

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

5.1.1. start

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

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

A valid JSEvent MUST include this property.

5.1.2. timeZone

Type: "String|null"

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

5.1.3. duration

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

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

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

5.1.4. isAllDay

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

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

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

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

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

5.1.5. status

Type: "String"

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

If set, it MUST be one of:

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

5.2. JSTask properties

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

5.2.1. due

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

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

5.2.2. start

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

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

5.2.3. timeZone

Type: "String|null"

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

5.2.4. estimatedDuration

Type: "Duration|null", e.g. "P2DT3H"

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

If the *start* and *due* properties are set, the estimated duration SHOULD be less than or equal to the time interval between these properties.

5.2.5. completed

Type: "UTCDate|null", e.g. "2016-06-13T12:00:00Z"

Specifies the date/time the task was completed.

If the task is recurring and has future instances, a client may want to denote a specific task recurrence as completed but leave other instances as uncompleted. One way to achieve this is by overriding the completed property in the task *recurrenceOverrides*. However, this could produce a long list of completion times for regularly recurring tasks. An alternative approach is to split the JSTask into a current, single instance of JSTask with this instance completion time and a future recurring instance. Also see the definition of the *relatedTo* on splitting.

5.2.6. isAllDay

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

Specifies if the task is an all day task.

If *isAllDay* is true, then the *start* and *due* properties MUST have a time component of "T00:00:00". Note that the *estimatedDuration* property MAY contain a non-zero time duration. All-day tasks MAY be bound to a specific time-zone, as defined by the *timeZone* property.

5.2.7. progress

In addition to the common properties of a *Participant* object (Section 4.4.5), a Participant within a JSTask supports the following property:

- *progress*: "ParticipantProgress|null" The progress of the participant for this task, if known.

A *ParticipantProgress* object has the following properties:

- *status*: "String" Describes the completion status of the participant's progress.

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

  - "completed": The progress of this participant is complete.

  - "in-process": The progress of this participant is in process.

- *timestamp*: "UTCDate" Describes the latest time when the participant progress got updated.

5.2.8. status

Type: "String"

The scheduling status (Section 4.4) of a JSTask defaults to "needs-action" if omitted.

If set, it MUST be one of:

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

- "completed": Indicates the task is completed. If this value is set, then the timestamp in the *completed* property MUST NOT be null.

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

- "cancelled": Indicates the task is cancelled.

5.3. JSGroup properties

JSGroup supports the following JSCalendar properties (Section 4):

- @type

- uid

- created

- updated

- categories

- keywords
- name
- description
- htmlDescription
- color
- links

as well as the following JSGroup-specific properties:

5.3.1. entries

Type: "(JSTask|JSEvent)[]|null"

A list of group members. The list MAY contain multiple object types and implementations MUST ignore entries of unknown type. The property value MUST either be "null" or the list MUST NOT be empty.

5.3.2. source

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

The source from which updated versions of this group may be retrieved from. If the value is not "null", it MUST be a URI.

6. Conversion from and to iCalendar

This section specifies which JSCalendar properties can be mapped from and to iCalendar format. Implementations SHOULD follow these conversion guidelines. Still, JSCalendar does not restrict itself to iCalendar and conversion between these two formats MAY be lossy.

6.1. JSEvent

The iCalendar counterpart to *JSEvent* is the VEVENT component type [RFC5545]. A VEVENT component that is a direct child of a VCALENDAR component is equivalent to a standalone JSEvent. A VEVENT component *within* a VEVENT maps to the entries of the JSEvent *recurrenceOverrides* property.

Property	iCalendar counterpart
isAllDay	True, if the type of the DTSTART property in iCalendar is DATE. When translating from JSCalendar the iCalendar DTSTART property is of DATE value type, if the *isAllDay* property is set to true and the *timeZone* property is null.
start	Corresponds to the DTSTART property in iCalendar. Note that time-zone information is stored separately in JSEvent.
timeZone	Corresponds to the TZID part of the DTSTART property in iCalendar. If the event has a different end time-zone to start time-zone, this should be added as a JSCalendar *location* with just a *timeZone* property and "rel="end"".
duration	Corresponds to the DURATION or DSTART+DTEND properties in iCalendar.

Table 2: Translation between JSEvent and iCalendar

6.2. JSTask

The iCalendar counterpart to *JSTask* is the VTODO component type [RFC5545]. A VTODO component that is a direct child of a VCALENDAR component is equivalent to a standalone JSTask. A VTODO component *within* a master VTODO maps to the entries of the JSTask *recurrenceOverrides* property.

Property	iCalendar counterpart
isAllDay	True, if the type of the DTSTART property in iCalendar is DATE. When translating from JSCalendar the iCalendar DTSTART property is of DATE value type, if the *isAllDay* property is set to true and the *timeZone* property is null.
due	Corresponds to the DUE and DTSTART+DURATION properties in iCalendar. When mapping iCalendar VTODOs with DTSTART+DURATION, the due date is the result of adding DURATION to DTSTART in the DTSTART time-zone.
start	Corresponds to the DTSTART property in iCalendar.
timeZone	Corresponds to the TZID part of the DTSTART/DUE properties in iCalendar. If the task has a different end time-zone to start or due time-zone, this should be added as a JSCalendar *location* with just a *timeZone* property and "rel="end"".
estimatedDuration	Corresponds to the ESTIMATED-DURATION iCalendar property. *NON-STANDARD*: this property is currently non-standard, see [draft-apthorp-ical-tasks].
completed	Maps to the COMPLETED iCalendar property.
progress	Corresponds to the PARTSTAT and COMPLETED properties in iCalendar.

Table 3: Translation between JSTask and iCalendar

6.3. JSGroup

A JSGroup converts to a iCalendar VCALENDAR containing VEVENT or VTODO components.

Property	iCalendar counterpart
entries	The VEVENT and VTODO components within a top-level VCALENDAR component.
source	Corresponds to the SOURCE property in iCalendar.

Table 4: Translation between JSGroup and iCalendar

6.4. Common properties

Property	iCalendar counterpart
alerts	An *Alert* corresponds to the VALARM component in iCalendar, where the *action* is determined by the iCalendar ACTION property value (e.g., a "DISPLAY" property indicates that the JSCalendar Alert action is a *DisplayAction* and similarly an iCalendar "EMAIL" value for *EmailAction* action). The *relativeTo* and *offset* properties corresponds to the iCalendar TRIGGER property. *NON-STANDARD*: The iCalendar properties for JSCalendar Alert actions are non-standard, see [draft-daboo-valarm-extensions].
categories	Corresponds to the STRUCTURED-CATEGORY property in iCalendar, see. *NON-STANDARD*: this property is currently non-standard, see [draft-ietf-calext-ical-relations].
color	Corresponds to the COLOR property in iCalendar, as specified in [RFC7986].
created	Corresponds to the CREATED property in iCalendar.
description	Corresponds to the DESCRIPTION property in iCalendar.
htmlDescription	There is no direct equivalent in iCalendar. If the *description* is empty, implementations SHOUD store a plain text

	version of *htmlDescription* in iCalendar DESCRIPTION.
freeBusyStatus	Corresponds to the TRANSP property in iCalendar.
keywords	Corresponds to the CATEGORIES property in iCalendar, as specified in [RFC7986].
links	Corresponds to the ATTACH ([RFC5545]) and IMAGE iCalendar properties ([RFC7986]).
locale	Corresponds to the LANGUAGE parameter in iCalendar, which is added to individual properties. When converting from iCalendar, one language must be picked as the main locale for the object, and all properties in other languages moved to the localizations JSEvent property.
localizations	Corresponds to the LANGUAGE parameter in iCalendar, which is added to individual properties. When converting from iCalendar, one language must be picked as the main locale for the object, and all properties in other languages moved to the localizations JSEvent property.
locations	See Section 6.5.
method	Corresponds to the METHOD property in iCalendar.
participants	See Section 6.5.
priority	Corresponds to the PRIORITY property in iCalendar.
privacy	Corresponds to the CLASS property in iCalendar.
prodId	Corresponds to the PRODID property in iCalendar.
recurrenceOverrides	Corresponds to the RDATE and EXDATE properties in iCalendar, plus VEVENT (for JSEvent) or VTODO (for JSTask) instances with a recurrence-id.

recurrenceRule	Corresponds to the RRULE property in iCalendar. See the property definition at section Section 4.3.1 how to map a RRULE value.
relatedTo	Corresponds to the RELATED-TO property in iCalendar.
replyTo	A *replyTo* property of type "imip" corresponds to the email address of the ORGANIZER property in iCalendar. There is no iCalendar representation for the "web" type.
sequence	Corresponds to the SEQUENCE property in iCalendar.
status	Corresponds to the STATUS property in iCalendar (converted to lower-case).
title	Corresponds to the SUMMARY property in iCalendar.
uid	Corresponds to the UID property in iCalendar.
updated	Corresponds to the DTSTAMP and LAST-MODIFIED properties in iCalendar. (These are only different in the iTIP case, and the difference is not actually useful.)

Table 5: Translation between JSCalendar and iCalendar

6.5. Locations and participants

Both JSCalendar participants and locations have counterparts in iCalendar but provide richer representation.

The following table outlines translation of JSCalendar participants. Where iCalendar has distinct properties for ORGANIZER and ATTENDEE, these are merged in JSCalendar into the Participant object type.

Property	iCalendar counterpart
name	the CN parameter
kind	the CUTYPE parameter
rsvpResponse	the PARTSTAT parameter
role	the ORGANIZER and ATTENDEE properties. Owners map to the iCalendar ORGANIZER property, where mapping multiple owners to iCalendar is implementation-specific but MUST be consistent across mappings of the same object.
participation	the ROLE parameter
locationId	the JSCalendar identifier of a mapped CONFERENCE property that has the MODERATOR feature defined in its FEATURE parameter values. If multiple such CONFERENCE properties are defined in iCalendar, then the one with the most interactive features is chosen (VIDEO over AUDIO over CHAT over anything else).
rsvpWanted	the RSVP parameter
delegatedTo	the DELEGATED-TO parameter
delegatedFrom	the DELEGATED-FROM parameter
memberOf	the MEMBER parameter
scheduleSequence	the SEQUENCE property of the participant's latest iMIP message
scheduleUpdated	the DTSTAMP property of the participant's latest iMIP message

Table 6: Translation of Participant between JSCalendar and iCalendar

For JSCalendar locations, the iCalendar counterparts are the [RFC5545] LOCATION and the extended iCalendar [RFC7986] CONFERENCE properties.

An iCalendar LOCATION property becomes a JSCalendar Location with just a description property. CONFERENCE property values in iCalendar

map to locations with *rel* type "virtual". The location *feature* property value corresponds to the extended iCalendar FEATURE property parameter values defined in [RFC7986]. Both iCalendar PHONE and AUDIO features map to the "audio" feature and the FEED parameter value is omitted. See the mapping for *locationId* in Table 6 on how to map CONFERENCE properties that contain the MODERATOR feature.

6.6. Unknown properties

Both JSCalendar and iCalendar calendar objects may contain properties that are not expressible in the other format. This specification does not mandate how to preserve these properties. Instead, it leaves negotiation on how to treat unknown properties to client and server implementations and their protocol used to exchange calendar objects.

Two notable options to represent and preserve arbitrary iCalendar object properties in JSCalendar are:

- *JCal*: Define iCalendar properties in JCal format ([RFC7265]) in a vendor-specific property of the JCalendar object. The JCal-formatted value may either only contain iCalendar properties that were not mapped to JSCalendar properties, or contain the complete iCalendar object representation.

- *Alternate link*: Define an alternate link (Section 4.2.5) value pointing to the iCalendar representation of the JSCalendar object. E.g. the alternative representation of a VEVENT would be represented as a link with rel "alternate" and type "text/calendar;component=VEVENT".

7. JSCalendar object examples

The following examples illustrate several aspects of the JSCalendar data model and format. The examples may omit mandatory or additional properties, which is indicated by a placeholder property with key "...". While most of the examples use calendar event objects, they are also illustrative for tasks.

7.1. Simple event

This example illustrates a simple one-time event. It specifies a one-time event that begins on January 15, 2018 at 1pm New York local time and ends after 1 hour.

```
{
  "@type": "jsevent",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f1",
  "updated": "2018-01-15T18:00:00Z",
  "title": "Some event",
  "start": "2018-01-15T13:00:00",
  "timeZone": "America/New_York",
  "duration": "PT1H"
}
```

7.2. Simple task

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

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

7.3. Simple group

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

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

7.4. All-day event

This example illustrates an event for an international holiday. It specifies an all-day event on April 1 that occurs every year since the year 1900.

```
{
  "...": "",
  "title": "April Fool's Day",
  "isAllDay": true,
  "start": "1900-04-01T00:00:00",
  "duration": "P1D",
  "recurrenceRule": {
    "frequency": "yearly"
  }
}
```

7.5. Task with a due date

This example illustrates a task with a due date. It is a reminder to buy groceries before 6pm Vienna local time on January 19, 2018. The calendar user expects to need 1 hour for shopping.

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

7.6. Event with end time-zone

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

```
{
  "...": "",
  "title": "Flight XY51 from FRA to NRT",
  "start": "2018-04-01T09:00:00",
  "timeZone": "Europe/Berlin",
  "duration": "PT10H30M",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "rel": "end",
      "timeZone": "Asia/Tokyo"
    }
  }
}
```

7.7. Floating-time event (with recurrence)

This example illustrates the use of floating-time. Since January 1, 2018, a calendar user blocks 30 minutes every day to practice Yoga at 7am local time, in whatever time-zone the user is located on that date.

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

7.8. Event with multiple locations and localization

This example illustrates an event that happens at both a physical and a virtual location. Fans can see a live convert on premises or online. The event title and descriptions are localized. (Note: the localization of the event description contains an UTF-8 encoded German Umlaut. This character may have been replaced with ASCII characters in the plain-text rendering of this RFC document)

```
{
  "...": "",
  "title": "Live from Music Bowl: The Band",
  "description": "Go see the biggest music event ever!",
  "locale": "en",
  "start": "2018-07-04T17:00:00",
  "timeZone": "America/New_York",
  "duration": "PT3H",
  "locations": {
    "9366e041-bb4c-4aa4-b249-b4657cab925c": {
      "name": "The Music Bowl",
      "description": "Music Bowl, Central Park, New York",
      "coordinates": "geo:40.7829,73.9654"
    },
    "6f3696c6-1e07-47d0-9ce1-f50014b0041a": {
      "name": "Free live Stream from Music Bowl",
      "rel": "virtual",
      "features": [
        "video",
        "audio",
        "chat"
      ],
      "uri": "https://stream.example.com/the_band_2018"
    }
  },
  "localizations": {
    "de": {
      "title": "Live von der Music Bowl: The Band!",
      "description": "Schau dir das groesste Musikereignis an!",
      "locations/6f3696c6-1e07-47d0-9ce1-f50014b0041a/name":
                              "Gratis Live-Stream aus der Music Bowl"
    }
  }
}
```

7.9. Recurring event with overrides

This example illustrates the use of recurrence overrides. A math course at a University is held for the first time on January 8, 2018 at 9am London time and occurs every week until June 25, 2018. Each lecture lasts for one hour and 30 minutes and is located at the Mathematics department. This event has exceptional occurrences: at the last occurrence of the course is an exam, which lasts for 2 hours and starts at 10am. Also, the location of the exam differs from the usual location. On April 2, May 7 and May 28 no course is held.

```
{
  "...": "",
  "title": "Calculus I",
  "start": "2018-01-08T09:00:00",
  "timeZone": "Europe/London",
  "duration": "PT1H30M",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "title": "Math lab room 1",
      "description": "Math Lab I, Department of Mathematics"
    }
  },
  "recurrenceRule": {
    "frequency": "weekly",
    "until": "2018-06-25T09:00:00"
  },
  "recurrenceOverrides": {
    "2018-04-02T09:00:00": null,
    "2018-05-07T09:00:00": null,
    "2018-05-28T09:00:00": null,
    "2018-06-25T09:00:00": {
      "title": "Calculus I Exam",
      "start": "2018-06-25T10:00:00",
      "duration": "PT2H",
      "locations": {
        "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
          "title": "Big Auditorium",
          "description": "Big Auditorium, Other Road"
        }
      }
    }
  }
}
```

7.10. Recurring event with participants

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

```
{
  "...": "",
  "title": "FooBar team meeting",
  "start": "2018-01-08T09:00:00",
  "timeZone": "Africa/Johannesburg",
  "duration": "PT1H",
  "locations": {
    "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
      "title": "ChatMe meeting room",
      "rel": "virtual",
      "features": [
        "audio",
        "chat",
        "video"
      ],
      "uri": "https://chatme.example.com?id=1234567"
    }
  },
  "recurrenceRule": {
    "frequency": "weekly"
  },
  "replyTo": {
    "imip": "zoe@foobar.example.com"
  },
  "participants": {
    "tom@foobar.example.com": {
      "name": "Tom Tool",
      "email": "tom@foobar.example.com",
      "rsvpResponse": "accepted",
      "roles": [
        "attendee"
      ]
    },
    "zoe@foobar.example.com": {
      "name": "Zoe Zelda",
      "email": "zoe@foobar.example.com",
      "rsvpResponse": "accepted",
      "roles": [
        "owner",
        "chair"
```

```
        ]
      },
      "...": ""
    },
    "recurrenceOverrides": {
      "2018-03-08T09:00:00": {
        "participants/tom@foobar.example.com/rsvpResponse": "declined"
      }
    }
  }
```

8. Security Considerations

The use of JSON as a format does have its own inherent security risks as discussed in Section 12 of [RFC8259]. Even though JSON is considered a safe subset of JavaScript, it should be kept in mind that a flaw in the parser processing JSON could still impose a threat, which doesn't arise with conventional iCalendar data.

With this in mind, a parser for JSON data aware of the security implications should be used for the format described in this document. For example, the use of JavaScript's "eval()" function is considered an unacceptable security risk, as described in Section 12 of[RFC8259]. A native parser with full awareness of the JSON format should be preferred.

9. IANA Considerations

This document amends the "application/calendar" MIME media type defined in [RFC7265].

New optional parameter: "type" with value being one of "jsevent", "jstask", "jsgroup". The parameter MUST NOT occur more than once.

10. Acknowledgments

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

11. References

11.1. Normative References

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

[RFC2183] Troost, R., Dorner, S., and K. Moore, Ed., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, DOI 10.17487/RFC2183, August 1997, <https://www.rfc-editor.org/info/rfc2183>.

[RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, DOI 10.17487/RFC2392, August 1998, <https://www.rfc-editor.org/info/rfc2392>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

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

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

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

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

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

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <https://www.rfc-editor.org/info/rfc5870>.

[RFC6047] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 6047, DOI 10.17487/RFC6047, December 2010, <https://www.rfc-editor.org/info/rfc6047>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

[RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", RFC 7265, DOI 10.17487/RFC7265, May 2014, <https://www.rfc-editor.org/info/rfc7265>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[RFC7529] Daboo, C. and G. Yakushev, "Non-Gregorian Recurrence Rules in the Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 7529, DOI 10.17487/RFC7529, May 2015, <https://www.rfc-editor.org/info/rfc7529>.

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

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

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

11.2. Informative References

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

[draft-daboo-valarm-extensions] "VALARM Extensions for iCalendar", <https://tools.ietf.org/html/draft-daboo-valarm-extensions>.

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

11.3. URIs

[1] https://www.w3.org/TR/html/

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

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

[4] http://www.iana.org/time-zones

Authors' Addresses

Neil Jenkins
FastMail
PO Box 234
Collins St West
Melbourne VIC 8007
Australia

Email: neilj@fastmailteam.com
URI: https://www.fastmail.com

Robert Stepanek
FastMail
PO Box 234
Collins St West
Melbourne VIC 8007
Australia

Email: rsto@fastmailteam.com
URI: https://www.fastmail.com