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

N. Jenkins
R. Stepanek
FastMail

June 29, 2019

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

Abstract

This specification defines a data model and JSON representation of calendar data that can be used for storage and data exchange in a calendaring and scheduling environment. It aims to be an alternative to the widely deployed iCalendar data format and to be unambiguous, extendable and simple to process. In contrast to the JSON-based jCal format, it is not a direct mapping from iCalendar and expands semantics where appropriate.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 31, 2019.

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] and [RFC7986] and extensions, but the specification should add new attributes or value types, or not support existing ones, where appropriate. Conversion between the data formats need not fully preserve semantic meaning.

- Extensions, such as new properties and components, MUST NOT lead to requiring an update to this document.

The representation of this data model is defined in the I-JSON format [RFC7493], which is a strict subset of the JavaScript Object Notation (JSON) Data Interchange Format [RFC8259]. Using JSON is mostly a pragmatic choice: its widespread use makes JSCalendar easier to adopt, and the ready availability of production-ready JSON implementations eliminates a whole category of parser-related interoperability issues.

1.1. Relation to the iCalendar format

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

For example, iCalendar defines various formats for local times, UTC time and dates, which confuses new users. Other sources for errors are the requirement for custom time zone definitions within a single calendar component, as well as the iCalendar format itself; the

latter causing interoperability issues due to misuse of CR LF terminated strings, line continuations and subtle differences between iCalendar parsers. Lastly, up until recently the iCalendar format did not have a way to express a concise difference between two calendar components, which results in verbose exchanges during scheduling.

1.2. Relation to the jCal format

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

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

1.3. Notational Conventions

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

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

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

2. JSCalendar objects

This section describes the calendar object types specified by JSCalendar.

2.1. JSEvent

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

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

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

2.2. JSTask

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

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

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

A JSTask may start and be due at certain points in time, may take some estimated time to complete and may recur; none of which is required. This notably differs from JSEvent (Section 2.1) which is required to start at a certain point in time and typically takes some non-zero duration to complete.

2.3. JSGroup

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

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

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

3. Structure of JSCalendar objects

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

The object has a collection of properties, as specified in the following sections. Properties are specified as being either mandatory or optional. Optional properties may have a default value, if explicitly specified in the property definition.

3.1. Type signatures

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

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

- "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. UTCDateTime

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

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

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

3.2.2. LocalDateTime

This is a date-time string _with no time zone/offset information_. It is otherwise in the same format as UTCDateTime, including fractional seconds. For example "2006-01-02T15:04:05" and "2006-01-02T15:04:05.003" are both valid. The time zone to associate the LocalDateTime with comes from an associated property, or if no time zone is associated it defines _floating time_. Floating date-times are not tied to any specific time zone. Instead, they occur in every time zone at the same wall-clock time (as opposed to the same instant point in time).

3.2.3. Duration

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

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

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

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

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

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

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

3.2.4. PatchObject

A PatchObject is of type "String[*|null]", and represents an unordered set of patches on a JSON object. The keys are a path in a subset of [RFC6901] JSON pointer format, with an implicit leading "/" (i.e. prefix each key with "/" before applying the JSON pointer evaluation algorithm).

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

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

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

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

The value associated with each pointer is either:

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

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

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

3.2.5. Identifiers

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

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

Nevertheless, a UUID typically is a good choice.

3.2.6. Time Zones

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

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

3.2.7. Normalization and equivalence

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

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

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

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

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

3.3. Custom property extensions and values

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

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

4. Common JSCalendar properties

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

4.1. Metadata properties

4.1.1. @type

Type: String (mandatory).

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

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

- "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), and the random or pseudo-random version is recommended.

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

4.1.3. relatedTo

Type: String[Relation] (optional).

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

A Relation object has the following properties:

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

  Keys in the set MUST be one of the following values, defined in a future specification or a vendor-specific value:

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

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

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

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

  The value for each key in the set MUST be "true".

If an object is split to make a "this and future" change to a recurrence, the original object MUST be truncated to end at the previous occurrence before this split, and a new object created to represent all the objects after the split. A "next" relation MUST be set on the original object's relatedTo property for the UID of the new object. A "first" relation for the UID of the first object in the series MUST be set on the new object. Clients can then follow these UIDs to get the complete set of objects if the user wishes to modify them all at once.

4.1.4. prodId

Type: String (optional).

The identifier for the product that created the JSCalendar object.

The vendor of the implementation SHOULD ensure that this is a globally unique identifier, using some technique such as an FPI value, as defined in [ISO.9070.1991]. It MUST only use characters of an iCalendar TEXT data value (see section 3.3.11 in [RFC5545]).

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

4.1.5. created

Type: UTCDateTime (optional).

The date and time this object was initially created.

4.1.6. updated

Type: UTCDateTime (mandatory).

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

4.1.7. sequence

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

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

4.1.8. method

Type: String (optional).

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

4.2. What and where properties

4.2.1. title

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

A short summary of the object.

4.2.2. description

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

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

4.2.3. descriptionContentType

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

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

4.2.4. showWithoutTime

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

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

4.2.5. locations

Type: String[Location] (optional).

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

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

- 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.
- relativeTo: String (optional). The relation type of this location to the JSCalendar object.

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

  * "start": The JSCalendar object starts at this location.
  * "end": The JSCalendar object ends at this location.
- timeZone: String (optional). A time zone for this location. Also see Section 3.2.6.
- coordinates: String (optional). An [RFC5870] "geo:" URI for the location.
- linkIds: String[Boolean] (optional). A set of link ids for links to alternate representations of this location. Each key in the set MUST be the identifier of a Link object defined in the "links" property of this calendar object. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

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

4.2.6. virtualLocations

Type: String[VirtualLocation] (optional).

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

A VirtualLocation object has the following properties.

- name: String (optional, default: empty String). The human-readable name of the virtual location.

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

- uri: String (mandatory). A URI that represents how to connect to this virtual location.

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

4.2.7. links

Type: String[Link] (optional).

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

A Link object has the following properties:

- href: String (mandatory). A URI from which the resource may be fetched.

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

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

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

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

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

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

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

  Links with a rel of "icon" SHOULD be considered by the client to be an image that it MAY use when presenting the calendar data to a user. The "display" property MAY be set to indicate the purpose of this image.

- display: String (optional). Describes the intended purpose of a link to an image. If set, the "rel" property MUST be set to "icon". The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:

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

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

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

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

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

4.2.8. locale

Type: String (optional).

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

4.2.9. keywords

Type: String[Boolean] (optional).

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

4.2.10. categories

Type: String[Boolean] (optional).

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

In contrast to keywords, categories typically are structured. For example, a vendor owning the domain "example.com" might define the categories "http://example.com/categories/sports/american-football"" and "http://example.com/categories/music/r-b".

4.2.11. color

Type: String (optional).

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

4.3. Recurrence properties

4.3.1. recurrenceRule

Type: Recurrence (optional).

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

A Recurrence object is a JSON object mapping of a RECUR value type in iCalendar, see [RFC5545] and[RFC7529]. A JSEvent recurs by applying the recurrence rule to the start date-time. A JSTask recurs by applying the recurrence rule to the start date-time, if defined, otherwise it recurs by the due date-time, if defined. If the task neither defines a start or due date-time, its "recurrenceRule" property value MUST be "null".

A Recurrence object has the following properties:

- frequency: String (mandatory). This MUST be one of the following values:

  * "yearly"

  * "monthly"

  * "weekly"

  * "daily"

  * "hourly"

  * "minutely"

  * "secondly"

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

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

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

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

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

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

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

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

- 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: LocalDateTime (optional). The UNTIL part from iCalendar. This MUST NOT be included if a "count" property is specified. Note: if not specified otherwise for a specific JSCalendar object, this date is presumed to be in the time zone specified in "timeZone". As in iCalendar, the until value bounds the recurrence rule inclusively.

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

1. A set of candidates is generated. This is every second within a period defined by the frequency property value:

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

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

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

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

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

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

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

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

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

* "secondly": the second itself, only.

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

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

   * byWeekNo: the date-time is in the nth week of the year. Negative numbers mean the nth last week of the year. This

corresponds to weeks according to week numbering as defined in ISO.8601.2004, with a week defined as a seven day period, starting on the firstDayOfWeek property value or Monday if omitted. Week number one of the calendar year is the first week that contains at least four days in that calendar year.

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

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

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

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

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

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

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

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

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

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

2. After applying the byMonthDay filter, if the day of the month is invalid for the given month and year, change the date to the first day of the next month (if skip == "forward") or the last day of the current month (if skip == "backward").

   3. If any valid date produced after applying the skip is already a candidate, eliminate the duplicate. (For example after adjusting, 30th February and 31st February would both become the same "real" date, so one is eliminated as a duplicate.)

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

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

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

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

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

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

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

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

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

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

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

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

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

* If frequency is "yearly" and no byYearDay property:

  + if there are no byMonth or byWeekNo properties, and either there is a byMonthDay property or there is no byDay property: Add a byMonth property with the sole value being the month of the start date-time.

  + if there is no byMonthDay, byWeekNo or byDay properties: Add a byMonthDay property with the sole value being the day-of-the-month of the start date-time.

  + if there is a byWeekNo property and no byMonthDay or byDay properties: Add a byDay property with the sole value being the day-of-the-week of the start date-time.

4.3.2. recurrenceOverrides

Type: LocalDateTime[PatchObject] (optional).

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

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

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

By default, an occurrence inherits all properties from the main object except the start (or due) date-time, which is shifted to the new start time of the LocalDateTime key. However, individual properties of the occurrence can be modified by a patch, or multiple patches. It is valid to patch the start property value, and this patch takes precedence over the LocalDateTime key. Both the

LocalDateTime key as well as the patched start date-time may occur before the original JSCalendar object's start or due date.

A pointer in the PatchObject MUST be ignored if it starts with one of the following prefixes:

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

4.3.3. excluded

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

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

4.4. Sharing and scheduling properties

4.4.1. priority

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

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

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

4.4.2. freeBusyStatus

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

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

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

4.4.3. privacy

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

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

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

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

- "public": The full details of the object are visible to those whom the object's calendar is shared with.
- "private": The details of the object are hidden; only the basic time and metadata is shared. The following properties MAY be shared, any other properties MUST NOT be shared:
  - @type
  - created
  - due

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

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

4.4.4. replyTo

Type: String[String] (optional).

Represents methods by which participants may submit their RSVP response to the organizer of the calendar object. The keys in the property value are the available methods and MUST only contain ASCII alphanumeric characters (A-Za-z0-9). The value is a URI to use that method. Future methods may be defined in future specifications; a calendar client MUST ignore any method it does not understand, but MUST preserve the method key and URI. This property MUST be omitted if no method is defined (rather than an empty object). If this property is set, the "participants" property of this calendar object MUST contain at least one participant.

The following methods are defined:

- "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.

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

4.4.5. participants

Type: String[Participant] (optional).

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

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

A Participant object has the following properties:

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

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

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

  The keys in the property value are the available methods and MUST only contain ASCII alphanumeric characters (A-Za-z0-9). The value is a URI to use that method. Future methods may be defined in future specifications; a calendar client MUST ignore any method it does not understand, but MUST preserve the method key and URI. This property MUST be omitted if no method is defined (rather than an empty object).

  The following methods are defined:

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

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

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

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

* "individual": a single person

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

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

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

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

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

  * "owner": The participant is an owner of the object.

  * "attendee": The participant is an attendee of the calendar object.

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

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

- locationId: String (optional). The location at which this participant is expected to be attending.

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

- participationStatus: String (optional, default: "needs-action"). The participation status, if any, of this participant.

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

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

  * "accepted": The invited participant will participate.

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

  * "tentative": The invited participant may participate.

- attendance: String (optional, default: "required"). The required attendance of this participant.

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

  * "none": Indicates a participant who is copied for information purposes only.

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

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

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

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

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

- scheduleUpdated: UTCDateTime (optional). The "updated" property of the last iMIP response from the participant.

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

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

- delegatedTo: String[Boolean] (optional). A set of participant ids that this participant has delegated their participation to. Each

key in the set MUST be the identifier of a participant. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

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

- memberOf: String[Boolean] (optional). A set of group participants that were invited to this calendar object, which caused this participant to be invited due to their membership of the group(s). Each key in the set MUST be the identifier of a participant. The value for each key in the set MUST be "true". This MUST be omitted if none (rather than an empty set).

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

4.5. Alerts properties

4.5.1. useDefaultAlerts

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

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

4.5.2. alerts

Type: String[Alert] (optional).

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

An Alert Object has the following properties:

- trigger: OffsetTrigger|UnknownTrigger. Defines when to trigger the alert.

An *OffsetTrigger* object has the following properties:

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

* offset: SignedDuration (mandatory). Defines to trigger the alert relative to the time property defined in the "relativeTo" property. If the calendar object does not define a time zone, the user's default time zone SHOULD be used when determining the offset, if known. Otherwise, the time zone to use is implementation specific.

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

  + "start": triggers the alert relative to the start of the calendar object

  + "end": triggers the alert relative to the end/due time of the calendar object

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

o acknowledged: UTCDateTime (optional).

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

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

o snoozed: UTCDateTime (optional).

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

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

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

  * "display": The alert should be displayed as appropriate for the current device and user context.

  * "email": The alert should trigger an email sent out to the user, notifying about the alert. This action is typically only appropriate for server implementations.

4.6. Multilingual properties

4.6.1. localizations

Type: String[PatchObject] (optional).

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

See the description of PatchObject (Section 3.2.4) for the structure of the PatchObject. The patches are applied to the top-level object. In addition to all the restrictions on patches specified there, the pointer also MUST NOT start with one of the following prefixes; any patch with a such a key MUST be ignored:

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

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

Note that this specification does not define how to maintain validity of localized content. For example, a client application changing a JSCalendar object's title property might also need to update any localizations of this property. Client implementations SHOULD provide the means to manage localizations, but how to achieve this is specific to the application's workflow and requirements.

4.7. Time zone properties

4.7.1. timeZones

Type: String[TimeZone] (optional).

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

- It MUST start with the "/" character (ASCII decimal 47; also see sections 3.2.19 of [RFC5545] and 3.6. of [RFC7808] for discussion of the forward slash character in time zone identifiers).
- It MUST be a valid "paramtext" value as specified in section 3.1. of [RFC5545].
- At least one other property in the same JSCalendar object MUST reference a time zone using this identifier (i.e. orphaned time zones are not allowed).

An identifier need only be unique to this JSCalendar object.

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

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

- lastModified: 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 set MUST be "true".

- standard: TimeZoneRule[] (optional). The STANDARD sub-components from iCalendar. The order MUST be preserved during conversion.

- daylight: TimeZoneRule[] (optional). The DAYLIGHT sub-components from iCalendar. The order MUST be preserved during conversion.

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

- start: LocalDateTime (mandatory). The DTSTART property from iCalendar.

- offsetTo: String (mandatory). The TZOFFSETTO property from iCalendar.

- offsetFrom: String (mandatory). The TZOFFSETFROM property from iCalendar.

- recurrenceRule: RecurrenceRule (optional). The RRULE property mapped as specified in Section 4.3.1. During recurrence rule evaluation, the "until" property value MUST be interpreted as a local time in the UTC time zone.

- recurrenceDates: LocalDateTime[Boolean] (optional). Maps the RDATE properties from iCalendar to a JSON set. The set is represented as an object, with the keys being the recurrence dates. The value for each key in the set MUST be "true".

- names: String[Boolean] (optional). Maps the TZNAME properties from iCalendar to a JSON set. The set is represented as an object, with the keys being the names. The value for each key in the set MUST be "true".

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

5. Type-specific JSCalendar properties

5.1. JSEvent properties

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

5.1.1. start

Type: LocalDateTime (mandatory).

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

5.1.2. timeZone

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

Identifies the time zone the event is scheduled in, or "null" for floating time. If omitted, this MUST be presumed to be "null" (i.e. floating time). Also see Section 3.2.6.

5.1.3. duration

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

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

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

5.1.4. status

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

The scheduling status (Section 4.4) of a JSEvent. If set, it MUST be one of:

- "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: LocalDateTime (optional).

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

5.2.2. start

Type: LocalDateTime (optional).

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

5.2.3. timeZone

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

Identifies the time zone the task is scheduled in, or "null" for floating time. If omitted, this MUST be presumed to be "null" (i.e. floating time). Also see Section 3.2.6.

5.2.4. estimatedDuration

Type: Duration (optional).

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

5.2.5. statusUpdatedAt

Type: UTCDateTime (optional).

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

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

5.2.6. progress

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

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

A ParticipantProgress object has the following properties:

- status: String (mandatory). Describes the completion status of the participant's progress.

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

  * "completed": The participant completed their task.

  * "in-process": The participant has started this task.

  * "failed": The participant failed to complete their task.

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

5.2.7. status

Type: String (optional).

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

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

If set, it MUST be one of:

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

5.3. JSGroup properties

JSGroup supports the following JSCalendar properties (Section 4):

- @type
- uid
- created
- updated
- categories

- keywords
- name
- description
- color
- links

as well as the following JSGroup-specific properties:

5.3.1. entries

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

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

5.3.2. source

Type: String (optional).

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

6. JSCalendar object examples

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

6.1. Simple event

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

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

6.2. Simple task

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

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

6.3. Simple group

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

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

6.4. All-day event

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

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

6.5. Task with a due date

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

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

6.6. Event with end time-zone

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

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

6.7. Floating-time event (with recurrence)

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

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

6.8. Event with multiple locations and localization

This example illustrates an event that happens at both a physical and a virtual location. Fans can see a live convert on premises or online. The event title and descriptions are localized.

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

6.9. Recurring event with overrides

This example illustrates the use of recurrence overrides. A math course at a University is held for the first time on January 8, 2018 at 9am London time and occurs every week until June 25, 2018. Each lecture lasts for one hour and 30 minutes and is located at the Mathematics department. This event has exceptional occurrences: at the last occurrence of the course is an exam, which lasts for 2 hours and starts at 10am. Also, the location of the exam differs from the usual location. On April 2 no course is held. On January 5 at 2pm is an optional introduction course, that occurs before the first regular lecture.

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

6.10. Recurring event with participants

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

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

```
    "start": "2018-01-08T09:00:00",
    "timeZone": "Africa/Johannesburg",
    "duration": "PT1H",
    "virtualLocations": {
      "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
        "name": "ChatMe meeting room",
        "uri": "https://chatme.example.com?id=1234567"
      }
    },
    "recurrenceRule": {
      "frequency": "weekly"
    },
    "replyTo": {
      "imip": "mailto:6489-4f14-a57f-c1@schedule.example.com"
    },
    "participants": {
      "dG9tQGZvb2Jhci5leGFtcGxlLmNvbQ": {
        "name": "Tom Tool",
        "email": "tom@foobar.example.com",
        "sendTo": {
          "imip": "mailto:6489-4f14-a57f-c1@calendar.example.com"
        },
        "participationStatus": "accepted",
        "roles": {
          "attendee": true
        }
      },
      "em9lQGZvb2Jhci5leGFtcGxlLmNvbQ": {
        "name": "Zoe Zelda",
        "email": "zoe@foobar.example.com",
        "sendTo": {
          "imip": "mailto:zoe@foobar.example.com"
        },
        "participationStatus": "accepted",
        "roles": {
          "owner": true,
          "attendee": true,
          "chair": true
        }
      },
      "...": ""
    },
    "recurrenceOverrides": {
     "2018-03-08T09:00:00": {
    "participants/dG9tQGZvb2Jhci5leGFtcGxlLmNvbQ/participationStatus":
                                                              "declined"
      }
    }
```

```
}
```

7. Security Considerations

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

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

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

8. IANA Considerations

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

Type name: application

Subtype name: jscalendar+json

Required parameters: type

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

Optional parameters: none

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

Security considerations: See Section 7 of this document.

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

Published specification: This specification.

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

Fragment identifier considerations: N/A

Additional information:

Magic number(s): N/A

File extensions(s): N/A

Macintosh file type code(s): N/A

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

Intended usage: COMMON

Restrictions on usage: N/A

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

Change controller: IETF

9. Acknowledgments

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

10. References

10.1. Normative References

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

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

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

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

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

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

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

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

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

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

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

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

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

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

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

[RFC7808] Douglass, M. and C. Daboo, "Time Zone Data Distribution Service", RFC 7808, DOI 10.17487/RFC7808, March 2016, <https://www.rfc-editor.org/info/rfc7808>.

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

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

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

10.2. Informative References

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

10.3. URIs

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

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

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

Authors' Addresses

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

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

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

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