JSCalendar: A JSON representation of calendar data
draft-ietf-calext-jscalendar-23

Abstract

This specification defines a data model and JSON representation of
calendar data that can be used for storage and data exchange in a
calendaring and scheduling environment. It aims to be an
alternative, and over time successor to, the widely deployed
iCalendar data format and to be unambiguous, extendable and simple to
process. In contrast to the 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 August 21, 2020.

Copyright Notice

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:

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

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

   o The data model should avoid ambiguities and make it difficult to
      make mistakes during implementation.

   o The data model should be compatible with the iCalendar data format
      [RFC5545] [RFC7986] and extensions, but the specification should
      add new attributes where the iCalendar format currently lacks
      expressivity, and drop widely unused, obsolete or redundant
      properties. This means translation with no loss of semantics

      should be easy with most common iCalendar files but is not
      guaranteed with the full specification.

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

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

1.1. Motivation and Relation to iCalendar and jCal

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

   For example, iCalendar defines various formats for local times, UTC
   time and dates, which confuses new users and often leads to
   implementation errors. 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.
   The definition of recurrence rules is ambiguous and has resulted in
   differing understandings even between experienced calendar
   developers.

   In recent years, many new products and services have appeared that
   wish to use a JSON representation of calendar data within their API.
   The JSON format for iCalendar data, jCal [RFC7265], is a direct
   mapping between iCalendar and JSON. In its effort to represent full
   iCalendar semantics, it inherits all the same pitfalls and uses a
   complicated JSON structure unlike most common JSON data
   representations.

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

a standard, elegant representation as an alternative to new
proprietary formats.

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.

1.3. Type Signatures

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

o "*" - The type is undefined (the value could be any type, although
   permitted values may be constrained by the context of this value).

o "String" - The JSON string type.

o "Number" - The JSON number type.

o "Boolean" - The JSON boolean type.

o "A[B]" - A JSON object where the keys are all of type "A", and the
   values are all of type "B".

o "A[]" - An array of values of type "A".

o "A|B" - The value is either of type "A" or of type "B".

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

1.4. Data Types

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

1.4.1. Int

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

1.4.2. UnsignedInt

   Where "UnsignedInt" is given as a data type, it means an "Int" where
   the value MUST be in the range 0 <= value <= 2^53-1.

1.4.3. UTCDateTime

   This is a string in [RFC3339] "date-time" format, with the further
   restrictions that any letters MUST be in uppercase, 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".

1.4.4. 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).

1.4.5. Duration

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

```
     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.

1.4.6. SignedDuration

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

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

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

1.4.7. Id

   Where "Id" is given as a data type, it means a "String" of at least 1
   and a maximum of 255 octets in size, and it MUST only contain
   characters from the "URL and Filename Safe" base64 alphabet, as
   defined in Section 5 of [RFC4648], excluding the pad character ("=").
   This means the allowed characters are the ASCII alphanumeric
   characters ("A-Za-z0-9"), hyphen ("-"), and underscore ("_").

   Unless otherwise specified, Ids are arbitrary and only have meaning
   within the object where they are being used. Ids need not be unique
   between different objects. For example, two JSEvent objects MAY use
   the same ids in their respective "links" properties. Or within the
   same JSEvent object the same Id could appear in the "participants"
   and "alerts" properties. This does not imply any semantic connection
   between the two.

   Nevertheless, a UUID is typically a good choice.

1.4.8. PatchObject

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

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

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

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

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

The value associated with each pointer is either:

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

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

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

1.4.9. Time Zones

By default, time zones in JSCalendar are identified by their name in
the IANA Time Zone Database [TZDB], 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 4.7.2).

1.4.10. Relation

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

o @type: "String" (mandatory)

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

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

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

   Keys in the set MUST be one of the following values, or specified
   in the property definition where the Relation object is used, or
   an IANA-registered value, or a vendor-specific value:

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

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

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

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

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

Note, the Relation object only has one property (except @type); it is
specified as an object with a single property to allow for extension
in the future.

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/or JSTask (Section 2.2) objects. Typically, objects are grouped by topic (e.g. by keywords) or calendar membership.

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

3. Structure of JSCalendar Objects

A JSCalendar object is a JSON object, 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. Normalization and Equivalence

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

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

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

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

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

3.2. Vendor-specific 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".

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

4. Common JSCalendar Properties

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

4.1. Metadata Properties

4.1.1. @type

Type: "String" (mandatory).

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

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

o "jstask": a JSCalendar task ([Section 2.2](#)).

o "jsgroup": a JSCalendar group ([Section 2.3](#)).

[4.1.2](#). uid

Type: "String" (mandatory).

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

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

[4.1.3](#). relatedTo

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

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

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

[4.1.4](#). prodId

Type: "String" (optional).

The identifier for the product that last updated the JSCalendar
object. This should be set whenever the data in the object is
modified (i.e., whenever the "updated" property is set).

The vendor of the implementation 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 of [RFC5545]).

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

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

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

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

4.1.8. method

Type: "String" (optional).

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

4.2. What and Where Properties

4.2.1. title

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

A short summary of the object.

4.2.2. description

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

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

4.2.3. descriptionContentType

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

   Describes the media type [RFC6838] of the contents of the
   "description" property. Media types MUST be sub-types of type
   "text", and SHOULD be "text/plain" or "text/html" [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.

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

4.2.5. locations

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

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

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

   o @type: "String" (mandatory)

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

o name: "String" (optional)

   The human-readable name of the location.

o description: "String" (optional)

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

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

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

o relativeTo: "String" (optional)

   The relation type of this location to the JSCalendar object.

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

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

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

o timeZone: "String" (optional)

   A time zone for this location. See also Section 1.4.9.

o coordinates: "String" (optional)

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

o linkIds: "Id[Boolean]" (optional)

   A set of link ids for links to alternate representations of this
   location. Each key in the set MUST be the id 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: "Id[VirtualLocation]" (optional).

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

A VirtualLocation object has the following properties.

o @type: "String" (mandatory)

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

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

   The human-readable name of the virtual location.

o description: "String" (optional)

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

o uri: "String" (mandatory)

   A URI that represents how to connect to this virtual location.

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

4.2.7. links

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

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

A Link object has the following properties:

o @type: "String" (mandatory)

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

o href: "String" (mandatory)

   A URI from which the resource may be fetched.

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

o cid: "String" (optional)

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

o contentType: "String" (optional)

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

o size: "UnsignedInt" (optional)

   The size, in octets, of the resource when fully decoded (i.e. the
   number of octets in the file the user would download), if known.

o rel: "String" (optional)

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

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

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

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

o display: "String" (optional)

   Describes the intended purpose of a link to an image. If set, the
   "rel" property MUST be set to "icon". The value MUST be either
   one of the following values, an IANA-registered value, or a
   vendor-specific value:

> * "badge": an image inline with the title of the object.
>
> * "graphic": a full image replacement for the object itself.
>
> * "fullsize": an image that is used to enhance the object.
>
> * "thumbnail": a smaller variant of "fullsize" to be used when
>   space for the image is constrained.

o title: "String" (optional)

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

4.2.8. locale

   Type: "String" (optional).

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

4.2.9. keywords

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

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

4.2.10. categories

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

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

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

4.2.11. color

   Type: "String" (optional).

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

RGB value in hexadecimal notation, as defined in Section 4.2.1 of CSS
Color Module Level 3.

4.3. Recurrence Properties

Some events and tasks occur at regular, or indeed irregular,
intervals. Rather than having to copy the data for every occurrence,
you can instead have a master event with a recurrence rule generating
the occurrences, and/or overrides that add extra dates or exceptions
to the rule.

4.3.1. recurrenceId

Type: "LocalDateTime" (optional).

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

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

4.3.2. recurrenceRules

Type: "RecurrenceRule[]" (optional).

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

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

A JSTask recurs by applying the recurrence rules to the "start" date-
time, if defined, otherwise it recurs by the "due" date-time, if
defined. If the task defines neither a "start" nor "due" date-time,
its "recurrenceRules" property value MUST be null.

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

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

o @type: "String" (mandatory)

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

o frequency: "String" (mandatory)

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

   * "yearly"

   * "monthly"

   * "weekly"

   * "daily"

   * "hourly"

   * "minutely"

   * "secondly"

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

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

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

   This is the INTERVAL part from iCalendar.

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

   The calendar system in which this recurrence rule operates, in
   lowercase. This MUST be either a CLDR-registered calendar system
   name, or a non-standard, experimental calendar system name
   prefixed with the characters "x-".

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

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

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

   * "omit"

   * "backward"

* "forward"

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

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

The day on which the week is considered to start, represented as a
lowercase abbreviated two-letter English day of the week. If
included, it MUST be one of the following values:

* "mo"

* "tu"

* "we"

* "th"

* "fr"

* "sa"

* "su"

This is the WKST part from iCalendar.

o byDay: "NDay[]" (optional)

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

* day: "String" (mandatory)

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

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

* nthOfPeriod: "Int" (optional)

   If present, rather than representing every occurrence of the
   weekday defined in the "day" property, it represents only a
   specific instance within the recurrence period. The value can
   be positive or negative, but MUST NOT be zero. A negative
   integer means nth-last of period.

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

o byMonthDay: "Int[]" (optional)

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

This is the BYMONTHDAY part in iCalendar.

o byMonth: "String[]" (optional)

The months in which to repeat. Each entry is a string
representation of a number, starting from "1" for the first month
in the calendar (e.g. "1" means January with the Gregorian
calendar), with an optional "L" suffix (see [RFC7529]) for leap
months (this MUST be uppercase, e.g. "3L"). The array MUST have
at least one entry if included.

This is the BYMONTH part from iCalendar.

o byYearDay: "Int[]" (optional)

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

This is the BYYEARDAY part from iCalendar.

o byWeekNo: "Int[]" (optional)

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

This is the BYWEEKNO part from iCalendar.

o byHour: "UnsignedInt[]" (optional)

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

   o byMinute: "UnsignedInt[]" (optional)

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

      This is the BYMINUTE part from iCalendar.

   o bySecond: "UnsignedInt[]" (optional)

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

      This is the BYSECOND part from iCalendar.

   o bySetPosition: "Int[]" (optional)

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

   o count: "UnsignedInt" (optional)

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

      This is the COUNT part from iCalendar.

   o until: "LocalDateTime" (optional)

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

      This is the UNTIL part from iCalendar.

4.3.2.1. Interpreting recurrence rules

   A recurrence rule specifies a set of date-times for recurring
   calendar objects. A recurrence rule has the following semantics.
   Note, wherever "year", "month" or "day of month" is used, this is

within the calendar system given by the "rscale" property, which
defaults to "gregorian" if omitted.

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

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

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

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

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

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

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

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

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

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

    * "secondly": the second itself, only.

2. Each date-time candidate is compared against all of the byX
    properties of the rule except bySetPosition. If any property in
    the rule does not match the date-time, 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 initial date-time to which the rule is applied (the "start"
    date-time for events; the "start" or "due" date-time for tasks)
    is always the first occurrence in the expansion (and is counted
    if the recurrence is limited by a "count" property), even if it
    would normally not match the rule.

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

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

* If frequency is not "secondly" and no bySecond property: Add a
  bySecond property with the sole value being the seconds value
  of the initial date-time.

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

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

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

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

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

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

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

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

4.3.3. recurrenceOverrides

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

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

If the recurrence id does not match a date-time from the recurrence
rule (or no rule is specified), it is to be treated as an additional
occurrence (like an RDATE from iCalendar). The patch object may
often be empty in this case.

If the patch object defines the "excluded" property 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 match
the recurrence id LocalDateTime. However, individual properties of
the occurrence can be modified by a patch, or multiple patches. It
is valid to patch the "start" property value, and this patch takes
precedence over the value generated from the recurrence id. Both the
recurrence id as well as the patched "start" date-time may occur
before the original JSCalendar object's "start" or "due" date.

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

o @type

o method

o privacy

o prodId

o recurrenceId

o recurrenceOverrides

o recurrenceRules

o relatedTo

o replyTo

o uid

4.3.4. excluded

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

Defines if this object is an overridden, excluded instance of a
recurring JSCalendar object (see Section 4.3.3). 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 included in the
occurrence expansion.

[4.4](#). **Sharing and Scheduling Properties**

[4.4.1](#). **priority**

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

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

   The priority is specified as an integer in the range 0 to 9. A value
   of 0 specifies an undefined priority. 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. This MUST be one of the following values, an IANA-
   registered value, or a vendor-specific value:

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

   o "busy": The object should be included when calculating whether the
      user is busy.

[4.4.3](#). **privacy**

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

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

   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, an IANA-
   registered value, or a vendor-specific value. Any value the client

or server doesn't understand should be preserved but treated as
equivalent to "private".

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

o "private": The details of the object are hidden; only the basic
   time and metadata is shared. The following properties MAY be
   shared, any other properties MUST NOT be shared:

   * @type

   * created

   * due

   * duration

   * estimatedDuration

   * freeBusyStatus

   * privacy

   * recurrenceOverrides. Only patches which apply to another
      permissible property are allowed to be shared.

   * sequence

   * showWithoutTime

   * start

   * timeZone

   * timeZones

   * uid

   * updated

o "secret": The object is hidden completely (as though it did not
   exist) when the calendar this object is in 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 and
registered with IANA; a calendar client MUST ignore any method it
does not understand, but MUST preserve the method key and URI. This
property MUST be omitted if no method is defined (rather than an
empty object). If this property is set, the "participants" property
of this calendar object MUST contain at least one participant.

The following methods are defined:

o   "imip": The organizer accepts an iMIP [RFC6047] response at this
    email address. The value MUST be a "mailto:" URI.

o   "web": Opening this URI in a web browser will provide the user
    with a page where they can submit a reply to the organizer.

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

4.4.5. participants

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

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

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

A Participant object has the following properties:

o   @type: "String" (mandatory)

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

o   name: "String" (optional)

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

o   email: "String" (optional)

      The email address for the participant.

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

      Represents methods by which the participant may receive the
      invitation and updates to the calendar object.

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

      The following methods are defined:

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

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

   o kind: "String" (optional)

      What kind of entity this participant is, if known.

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

      * "individual": a single person

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

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

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

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

      A set of roles that this participant fulfills.

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

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

*   "attendee": The participant is expected to attend.

*   "optional": The participant is invited but not required.

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

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

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

o locationId: "String" (optional)

   The location at which this participant is expected to be
   attending.

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

o language: "String" (optional)

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

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

   The participation status, if any, of this participant.

   The value MUST be either one of the following values, an IANA-
   registered value, 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.

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

o participationComment: "String" (optional)

   A note from the participant to explain their participation status.

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

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

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

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

   The value MUST be either one of the following values, an IANA-
   registered value, or a vendor-specific value:

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

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

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

o scheduleSequence: "UnsignedInt" (optional, default: 0)

   The sequence number of the last response from the participant. If
   defined, this MUST be a non-negative integer.

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

o   scheduleUpdated: "UTCDateTime" (optional)

    The "updated" property of the last iTIP response from the
    participant.

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

o   invitedBy: "String" (optional)

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

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

    A set of participant ids that this participant has delegated their
    participation to. Each key in the set MUST be the id of a
    participant. The value for each key in the set MUST be true.
    This MUST be omitted if none (rather than an empty set).

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

    A set of participant ids that this participant is acting as a
    delegate for. Each key in the set MUST be the id of a
    participant. The value for each key in the set MUST be true.
    This MUST be omitted if none (rather than an empty set).

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

    A set of group participants that were invited to this calendar
    object, which caused this participant to be invited due to their
    membership of the group(s). Each key in the set MUST be the id of
    a participant. The value for each key in the set MUST be true.
    This MUST be omitted if none (rather than an empty set).

o   linkIds: "String[Boolean]" (optional)

    A set of links to more information about this participant, for
    example in vCard format. The keys in the set MUST be the id 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).

o   progress: "String" (optional). This is only allowed if the
    Participant is part of a JSTask. It represents the progress of
    the participant for this task, if known. This property MUST NOT
    be set if the "participationStatus" of this participant is any

      value other than "accepted". See Section 5.2.4 for allowed values
      and semantics.

   o progressUpdated: "UTCDateTime" (optional). Specifies the date-
      time the progress property was last set on this participant. This
      is only allowed if the Participant is part of a JSTask.

4.5. Alerts Properties

4.5.1. useDefaultAlerts

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

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

4.5.2. alerts

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

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

      An Alert Object has the following properties:

   o @type: "String" (mandatory)

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

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

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

      An *OffsetTrigger* object has the following properties:

      * @type: "String" (mandatory)

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

      * offset: "SignedDuration" (mandatory).

Defines the offset at which to trigger the alert relative to
the time property defined in the "relativeTo" property of the
alert. Negative durations signify alerts before the time
property, positive durations signify alerts after. 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 that the alert offset is relative
   to. The value MUST be one of:

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

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

An *AbsoluteTrigger* object has the following properties:

* @type: "String" (mandatory)

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

* when: "UTCDateTime" (mandatory).

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

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

o acknowledged: "UTCDateTime" (optional)

   This records when an alert was last acknowledged. This is set
   when the user has dismissed the alert; other clients that sync
   this property SHOULD automatically dismiss or suppress duplicate
   alerts (alerts with the same alert id that triggered on or before
   this date-time).

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

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

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

Relates this alert to other alerts in the same JSCalendar object.
If the user wishes to snooze an alert, the application SHOULD
create an alert to trigger after snoozing. All snooze alerts
SHOULD set a relation to the identifier of the original alert.
The Relation object SHOULD set the "parent" relation type, but MAY
be empty.

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

Describes how to alert the user.

The value MUST be at most one of the following values, an IANA-
registered value, 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 1.4.8) for the structure
of the PatchObject. The patches are applied to the top-level
calendar object. In addition, the "locale" property of the patched
object is set to the language tag. All pointers for patches MUST end
with one of the following suffixes; any patch that does not follow
this MUST be ignored unless otherwise specified in a future RFC:

o title

o description

o name

A patch MUST NOT have the prefix "recurrenceOverrides"; any
localization of the override MUST be a patch to the localizations
property inside the override instead. For example, a patch to
"locations/abcd1234/title" is permissible, but a patch to "uid" or
"recurrenceOverrides/2018-01-05T14:00:00/title" is not.

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

4.7. Time Zone Properties

4.7.1. timeZone

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

Identifies the time zone the object is scheduled in, or null for
floating time. This is either a name from the IANA Time Zone
Database [TZDB] or the id of a custom time zone from the "timeZones"
property (see Section 1.4.9). If omitted, this MUST be presumed to
be null (i.e., floating time).

4.7.2. timeZones

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

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

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

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

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

An identifier need only be unique to this JSCalendar object.

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

o @type: "String" (mandatory)

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

o tzId: "String" (mandatory).

   The TZID property from iCalendar.

o lastModified: "UTCDateTime" (optional)

   The LAST-MODIFIED property from iCalendar.

o url: "String" (optional)

   The TZURL property from iCalendar.

o validUntil: "UTCDateTime" (optional)

   The TZUNTIL property from iCalendar specified in [RFC7808].

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

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

o standard: "TimeZoneRule[]" (optional)

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

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

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

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

o @type: "String" (mandatory)

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

   o start: "LocalDateTime" (mandatory)

      The DTSTART property from iCalendar.

   o offsetTo: "String" (mandatory)

      The TZOFFSETTO property from iCalendar.

   o offsetFrom: "String" (mandatory)

      The TZOFFSETFROM property from iCalendar.

   o recurrenceRule: "RecurrenceRule" (optional)

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

   o recurrenceDates: "LocalDateTime[Boolean]" (optional)

      Maps the RDATE properties from iCalendar to a JSON set. The set
      is represented as an object, with the keys being the recurrence
      dates. The value for each key in the set MUST be true.

   o names: "String[Boolean]" (optional)

      Maps the TZNAME properties from iCalendar to a JSON set. The set
      is represented as an object, with the keys being the names. The
      value for each key in the set MUST be true.

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

5. Type-specific JSCalendar Properties

5.1. JSEvent Properties

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

5.1.1. start

   Type: "LocalDateTime" (mandatory).

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

5.1.2. duration

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

   The zero or positive duration of the event in the event's start time
   zone.

   Note that a duration specified using weeks or days does not always
   correspond to an exact multiple of 24 hours. The number of
   hours/minutes/seconds may vary if it overlaps a period of
   discontinuity in the event's time zone, for example a change from
   standard time to daylight-savings time. Leap seconds MUST NOT be
   considered when 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. These semantics match the
   iCalendar DURATION value type ([RFC5545], Section 3.3.6).

   A JSEvent MAY involve start and end locations that are in different
   time zones (e.g. a trans-continental flight). This can be expressed
   using the "relativeTo" and "timeZone" properties of the JSEvent's
   Location objects (see Section 4.2.5).

5.1.3. status

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

   The scheduling status (Section 4.4) of a JSEvent. If set, it MUST be
   one of the following values, an IANA-registered value, or a vendor-
   specific value:

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

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

   o "tentative": Indicates the event may happen.

5.2. JSTask Properties

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

5.2.1. due

   Type: "LocalDateTime" (optional).

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

[5.2.2](#). **start**

Type: "LocalDateTime" (optional).

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

[5.2.3](#). **estimatedDuration**

Type: "Duration" (optional).

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

[5.2.4](#). **progress**

Type: "String" (optional).

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

o "completed": if the "progress" property value of all participants is "completed".

o "failed": if at least one "progress" property value of a participant is "failed".

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

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

If set, it MUST be one of the following values, an IANA-registered value, or a vendor-specific value:

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

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

o "completed": Indicates the task is completed.

o "failed": Indicates the task failed.

o "cancelled": Indicates the task was cancelled.

5.2.5. progressUpdated

   Type: "UTCDateTime" (optional).

   Specifies the date/time the "progress" property of either the task
   overall (Section 5.2.4) or a specific participant (Section 4.4.5) was
   last updated.

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

5.3. JSGroup Properties

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

   o @type

   o categories

   o color

   o created

   o description

   o descriptionContentType

   o keywords

   o links

   o locale

   o prodId

   o title

   o updated

   o uid

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

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. 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": {
       "2a358cee-6489-4f14-a57f-c104db4dc2f1": {
         "@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"
       },
       "2a358cee-6489-4f14-a57f-c104db4dc2f2": {
         "@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",
     "recurrenceRules": [{
       "@type": "RecurrenceRule",
       "frequency": "yearly"
     }]
   }
```

6.5. Task with a due date

   This example illustrates a task with a due date. It is a reminder to
   buy groceries before 6pm Vienna local time on January 19, 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": {
         "@type": "Location",
         "rel": "start",
         "name": "Frankfurt Airport (FRA)"
       },
       "c2c7ac67-dc13-411e-a7d4-0780fb61fb08": {
         "@type": "Location",
         "rel": "end",
         "name": "Narita International Airport (NRT)",
         "timeZone": "Asia/Tokyo"
       }
     }
   }
```

6.7. Floating-time event (with recurrence)

This example illustrates the use of floating-time. Since January 1, 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",
     "recurrenceRules": [{
       "@type": "RecurrenceRule",
       "frequency": "daily"
     }]
   }
```

6.8. Event with multiple locations and localization

This example illustrates an event that happens at both a physical and a virtual location. Fans can see a live 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": {
         "@type": "Location",
         "name": "The Music Bowl",
         "description": "Music Bowl, Central Park, New York",
         "coordinates": "geo:40.7829,73.9654"
       }
     },
     "virtualLocations": {
       "6f3696c6-1e07-47d0-9ce1-f50014b0041a": {
         "@type": "VirtualLocation",
         "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": {
         "@type": "Location",
         "title": "Math lab room 1",
         "description": "Math Lab I, Department of Mathematics"
       }
     },
     "recurrenceRules": [{
       "@type": "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": {
             "@type": "Location",
             "title": "Big Auditorium",
             "description": "Big Auditorium, Other Road"
           }
         }
       }
     }
   }
```

6.10. Recurring event with participants

   This example illustrates scheduled events. A team meeting occurs
   every week since January 8, 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": {
      "@type": "VirtualLocation",
      "name": "ChatMe meeting room",
      "uri": "https://chatme.example.com?id=1234567"
    }
  },
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "weekly"
  }],
  "replyTo": {
    "imip": "mailto:6489-4f14-a57f-c1@schedule.example.com"
  },
  "participants": {
    "dG9tQGZvb2Jhci5xlLmNvbQ": {
      "@type": "Participant",
      "name": "Tom Tool",
      "email": "tom@foobar.example.com",
      "sendTo": {
        "imip": "mailto:6489-4f14-a57f-c1@calendar.example.com"
      },
      "participationStatus": "accepted",
      "roles": {
        "attendee": true
      }
    },
    "em9lQGZvb2GFtcGxlLmNvbQ": {
      "@type": "Participant",
      "name": "Zoe Zelda",
      "email": "zoe@foobar.example.com",
      "sendTo": {
        "imip": "mailto:zoe@foobar.example.com"
      },
      "participationStatus": "accepted",
      "roles": {
        "owner": true,
        "attendee": true,
        "chair": true
      }
    },
    "...": ""
```

```
      },
      "recurrenceOverrides": {
        "2018-03-08T09:00:00": {
          "participants/dG9tQGZvb2Jhci5xlLmNvbQ/participationStatus":
                                                      "declined"
        }
      }
    }
```

7. Security Considerations

Calendaring and scheduling information is very privacy-sensitive.
The transmission of such information must be careful to protect it
from possible threats, such as eavesdropping, replay, message
insertion, deletion, modification, and man-in-the-middle attacks.
This document just defines the data format; such considerations are
primarily the concern of the API or method of storage and
transmission of such files.

7.1. Expanding Recurrences

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

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

7.2. JSON Parsing

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

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

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

characters or adjacent to the end of data (as in the previous
paragraph).

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

7.3. URI Values

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

8. IANA Considerations

8.1. Media Type Registration

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

8.2. Creation of "JSCalendar Properties" Registry

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

   This registry follows the Expert Review process ([RFC8126],
   Section 4.5) unless the "intended use" field is "common", in which
   case registration follows the Specification Required process
   ([RFC8126], Section 4.6). Preliminary community review for this
   registry is optional but strongly encouraged.

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

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

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

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

8.2.1. Preliminary Community Review

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

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

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

8.2.2. Submit Request to IANA

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

8.2.3. Designated Expert Review

The primary concern of the designated expert (DE) is preventing name
collisions and encouraging the submitter to document security and
privacy considerations. For a common-use registration, the DE is
expected to confirm that suitable documentation, as described in
Section 4.6 of [RFC8126], is available to ensure interoperability.
This preferably takes the form of an RFC, but for simple definitions
a description in the registry may be sufficient. The DE should also
verify that the property name does not conflict with work that is
active or already published within the IETF. A published
specification is not required for reserved or obsolete registrations.

Before a period of 30 days has passed, the DE will either approve or
deny the registration request and publish a notice of the decision to
the Calext WG mailing list or its successor, as well as inform IANA.
A denial notice must be justified by an explanation, and, in the

cases where it is possible, concrete suggestions on how the request
can be modified so as to become acceptable should be provided.

If the DE does not respond within 30 days, the registrant may request
the IESG take action to process the request in a timely manner.

8.2.4. Change Procedures

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

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

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

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

The IESG may reassign responsibility for a JSCalendar property. The
most common case of this will be to enable changes to be made to a
registration where the author of the registration has died, moved out
of contact, or is otherwise unable to make changes that are important
to the community.

8.2.5. JMAP Properties Registry Template

o Property Name: The name of the property. The property name MUST
   NOT already be registered for any of the objects listed in
   Context. Objects not listed in Context MAY already have
   registered a different property with the same name.

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

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

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

   o Intended Use: Common, reserved, or obsolete.

   o Change Controller: ("IETF" for IETF-stream RFCs).

8.2.6. Initial Contents for the JSCalendar Properties Registry

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

Property Name	Property Type	Property Context	Reference or Description
@type	String	JSEvent, JSTask, JSGroup, AbsoluteTrigger, Alert, Link, Location, OffsetTrigger, Participant, RecurrenceRule, Relation, TimeZone, VirtualLocation	Section 4.1.1, Section 4.5.2, Section 4.2.7, Section 4.2.5, Section 4.4.5, Section 4.3.2, Section 4.1.3, Section 4.7.2, Section 4.2.6
acknowledged	UTCDateTime	Alert	Section 4.5.2
action	String	Alert	Section 4.5.2
alerts	Id[Alert]	JSEvent, JSTask	Section 4.5.2

byDay	NDay[]	Recurrence Rule	Section 4.3.2
byHour	UnsignedInt[]	Recurrence Rule	Section 4.3.2
byMinute	UnsignedInt[]	Recurrence Rule	Section 4.3.2
byMonth	String[]	Recurrence Rule	Section 4.3.2
byMonthDay	Int[]	Recurrence Rule	Section 4.3.2
bySecond	UnsignedInt[]	Recurrence Rule	Section 4.3.2
bySetPosition	Int[]	Recurrence Rule	Section 4.3.2
byWeekNo	Int[]	Recurrence Rule	Section 4.3.2
byYearDay	Int[]	Recurrence Rule	Section 4.3.2
categories	String[Boolean]	JSEvent, JSTask, JSGroup	Section 4.2.10
cid	String	Link	Section 4.2.7
color	String	JSEvent, JSTask, JSGroup	Section 4.2.11
contentType	String	Link	Section 4.2.7
coordinates	String	Location	Section 4.2.5
count	UnsignedInt	Recurrence Rule	Section 4.3.2

created	UTCDateTime	JSEvent, JSTask, JSGroup	Section 4.1.5
delegatedFrom	String[Boolean]	Participant	Section 4.4.5
delegatedTo	String[Boolean]	Participant	Section 4.4.5
description	String	JSEvent, JSTask, Location, VirtualLocation	Section 4.2.2, Section 4.2.5, Section 4.2.6
descriptionContentType	String	JSEvent, JSTask	Section 4.2.3
display	String	Link	Section 4.2.7
due	LocalDateTime	JSTask	Section 5.2.1
duration	Duration	JSEvent	Section 5.1.2
email	String	Participant	Section 4.4.5
entries	String[JSTask|JSEvent]	JSGroup	Section 5.3.1
estimatedDuration	Duration	JSTask	Section 5.2.3
excluded	Boolean	JSEvent, JSTask	Section 4.3.4
expectReply	Boolean	Participant	Section 4.4.5
firstDayOfWeek	String	Recurrence Rule	Section 4.3.2
freeBusyStatu	String	JSEvent,	Section

s			JSTask	4.4.2
frequency	String		Recurrence Rule	Section 4.3.2
href	String		Link	Section 4.2.7
interval	UnsignedInt		Recurrence Rule	Section 4.3.2
invitedBy	String		Participant	Section 4.4.5
keywords	String[Boolean]		JSEvent, JSTask, JSGroup	Section 4.2.9
kind	String		Participant	Section 4.4.5
language	String		Participant	Section 4.4.5
linkIds	Id[Boolean]		Location, Participant	Section 4.2.5, Section 4.4.5
links	Id[Link]		JSGroup, JSEvent, JSTask	Section 4.2.7
locale	String		JSGroup, JSEvent, JSTask	Section 4.2.7
localizations	String[PatchObject]		JSEvent, JSTask	Section 4.6.1
locationId	String		Participant	Section 4.4.5
locations	Id[Location]		JSEvent, JSTask	Section 4.2.5
locationTypes	String[Boolean]		Location	Section 4.2.5

memberOf	String[Boolean]	Participant	Section 4.4.5	
method	String	JSEvent, JSTask	Section 4.1.8	
name	String	Location, VirtualLocation, Participant	Section 4.2.5, Section 4.2.6, Section 4.4.5	
offset	SignedDuration	OffsetTrigger	Section 4.5.2	
participants	Id[Participant]	JSEvent, JSTask	Section 4.4.5	
participationComment	String	Participant	Section 4.4.5	
participationStatus	String	Participant	Section 4.4.5	
priority	Int	JSEvent, JSTask	Section 4.4.1	
privacy	String	JSEvent, JSTask	Section 4.4.3	
prodId	String	JSEvent, JSTask, JSGroup	Section 4.1.4	
progress	String	JSTask, Participant	Section 5.2.4	
progressUpdated	UTCDateTime	JSTask, Participant	Section 5.2.5	
recurrenceId	LocalDateTime	JSEvent, JSTask	Section 4.3.1	
recurrenceOverrides	LocalDateTime[PatchObject]	JSEvent, JSTask	Section 4.3.3	

recurrenceRules	RecurrenceRule[]	JSEvent, JSTask	Section 4.3.2
rel	String	Link	Section 4.2.7
relatedTo	String[Relation]	JSEvent, JSTask, Alert	Section 4.1.3, Section 4.5.2
relation	String[Boolean]	Relation	Section 1.4.10
relativeTo	String	OffsetTrigger, Location	Section 4.5.2, Section 4.2.5
replyTo	String[String]	JSEvent, JSTask	Section 4.4.4
roles	String[Boolean]	Participant	Section 4.4.5
rscale	String	Recurrence Rule	Section 4.3.2
scheduleAgent	String	Participant	Section 4.4.5
scheduleSequence	UnsignedInt	Participant	Section 4.4.5
scheduleUpdated	UTCDateTime	Participant	Section 4.4.5
sendTo	String[String]	Participant	Section 4.4.5
sequence	UnsignedInt	JSEvent, JSTask	Section 4.1.7
showWithoutTime	Boolean	JSEvent, JSTask	Section 4.2.4
size	UnsignedInt	Link	Section 4.2.7

skip	String		Recurrence Rule	Section 4.3.2
source	String		JSGroup	Section 5.3.2
start	LocalDateTime		JSEvent, JSTask	Section 5.1.1, Section 5.2.2
status	String		JSEvent	Section 5.1.3
timeZone	String|null		JSEvent, JSTask, Location	Section 4.7.1, Section 4.2.5
timeZones	String[TimeZone]		JSEvent, JSTask	Section 4.7.2
title	String		JSEvent, JSTask, JSGroup, Link	Section 4.2.1
trigger	OffsetTrigger|AbsoluteTrigger|UnknownTrigger		Alert	Section 4.5.2
uid	String		JSEvent, JSTask, JSGroup	Section 4.1.2
until	LocalDateTime		Recurrence Rule	Section 4.3.2
updated	UTCDateTime		JSEvent, JSTask, JSGroup	Section 4.1.6
uri	String		VirtualLocation	Section 4.2.6
useDefaultAlerts	Boolean		JSEvent, JSTask	Section 4.5.1

virtualLocati	Id[VirtualLocation]	JSEvent,	Section
ons		JSTask	4.2.6
when	UTCDateTime	AbsoluteTr	Section
		igger	4.5.2

                               Table 1

8.3. Creation of "JSCalendar Types" Registry

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

8.3.1. JMAP Types Registry Template

   o Type Name: The name of the type.

   o Reference or Description: A brief description or RFC number and
      section reference where the Type is specified (may be omitted for
      "reserved" type names).

   o Intended Use: Common, reserved, or obsolete.

   o Change Controller: ("IETF" for IETF-stream RFCs).

8.3.2. Initial Contents for the JSCalendar Types Registry

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

Type Name	Reference or Description
Alert	Section 4.5.2
Boolean	Section 1.3
Duration	Section 1.4.5
Id	Section 1.4.7
Int	Section 1.4.1
LocalDateTime	Section 1.4.4
Link	Section 4.2.7
Location	Section 4.2.5
Number	Section 1.3
Participant	Section 4.4.5
PatchObject	Section 1.4.8
RecurrenceRule	Section 4.3.2
Relation	Section 1.4.10
SignedDuration	Section 1.4.6
String	Section 1.3
TimeZone	Section 4.7.2
TimeZoneRule	Section 4.7.2
UnsignedInt	Section 1.4.2
UTCDateTime	Section 1.4.3
VirtualLocation	Section 4.2.6

Table 2

8.4. **Creation of "JSCalendar Enum Values" Registry**

   The IANA will create the "JSCalendar Enum Values" registry to allow
   interoperable extension of semantics for properties with enumerable
   values. Each such property will have a subregistry of allowed
   values. The registration process for creating a new subregistry is
   the same as for the JSCalendar Properties registry. The registration
   process for a new enum value is the same but is only subject to
   Expert Review; a specification is not required for a new allowed
   value in an existing enum property where a simple description will
   suffice.

8.4.1. **JMAP Enum Subregistry Creation Template**

   This template is for adding a new subregistry to the JMAP Enum
   registry.

   o Registry Name: This MUST be of the form "Enum Values for
      {property-name} (Context: {context})" where:

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

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

   o Change Controller: ("IETF" for IETF-stream RFCs).

   o Experts: The initial list of experts for Expert Review of updates
      to the subregistry.

   o Initial Contents: The initial list of defined values for this
      enum, using the template defined in Section 8.4.2.

8.4.2. **JMAP Enum Subregistry Template**

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

   o Enum Value: The verbatim value of the enum.

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

8.4.3. Initial Contents for the JSCalendar Enum Registry

For each subregistry created in this section, all RFC section references are for this document, the change controller is IETF and the experts are Neil Jenkins and Robert Stepanek.

Enum Values for action (Context: Alert)

```
         +------------+--------------------------+
         | Enum Value | Reference or Description |
         +------------+--------------------------+
         | display    | Section 4.5.2            |
         |            |                          |
         | email      | Section 4.5.2            |
         +------------+--------------------------+
```

                          Table 3

Enum Values for display (Context: Link)

```
         +------------+--------------------------+
         | Enum Value | Reference or Description |
         +------------+--------------------------+
         | badge      | Section 4.2.7            |
         |            |                          |
         | graphic    | Section 4.2.7            |
         |            |                          |
         | fullsize   | Section 4.2.7            |
         |            |                          |
         | thumbnail  | Section 4.2.7            |
         +------------+--------------------------+
```

                          Table 4

Enum Values for freeBusyStatus (Context: JSEvent, JSTask)

```
             +------------+--------------------------+
             | Enum Value | Reference or Description |
             +------------+--------------------------+
             | free       | Section 4.4.2            |
             |            |                          |
             | busy       | Section 4.4.2            |
             +------------+--------------------------+
```

                              Table 5

Enum Values for kind (Context: Participant)

```
             +------------+--------------------------+
             | Enum Value | Reference or Description |
             +------------+--------------------------+
             | individual | Section 4.4.5            |
             |            |                          |
             | group      | Section 4.4.5            |
             |            |                          |
             | resource   | Section 4.4.5            |
             |            |                          |
             | location   | Section 4.4.5            |
             +------------+--------------------------+
```

                              Table 6

Enum Values for participationStatus (Context: Participant)

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

                              Table 7

   Enum Values for privacy (Context: JSEvent, JSTask)

               +------------+--------------------------+
               | Enum Value | Reference or Description |
               +------------+--------------------------+
               | public | Section 4.4.3 |
               | | |
               | private | Section 4.4.3 |
               | | |
               | secret | Section 4.4.3 |
               +------------+--------------------------+

                              Table 8

   Enum Values for progress (Context: JSTask, Participant)

               +--------------+-------------------------+
               | Enum Value | Reference or Description |
               +--------------+-------------------------+
               | needs-action | Section 5.2.4 |
               | | |
               | in-process | Section 5.2.4 |
               | | |
               | completed | Section 5.2.4 |
               | | |
               | failed | Section 5.2.4 |
               | | |
               | cancelled | Section 5.2.4 |
               +--------------+-------------------------+

                              Table 9

   Enum Values for relation (Context: Relation)

```
            +------------+--------------------------+
            | Enum Value | Reference or Description |
            +------------+--------------------------+
            | first      | Section 1.4.10           |
            |            |                          |
            | next       | Section 1.4.10           |
            |            |                          |
            | child      | Section 1.4.10           |
            |            |                          |
            | parent     | Section 1.4.10           |
            +------------+--------------------------+
```

                              Table 10

   Enum Values for relativeTo (Context: OffsetTrigger, Location)

```
            +------------+--------------------------+
            | Enum Value | Reference or Description |
            +------------+--------------------------+
            | start      | Section 4.5.2            |
            |            |                          |
            | end        | Section 4.5.2            |
            +------------+--------------------------+
```

                              Table 11

   Enum Values for roles (Context: Participant)

```
            +---------------+--------------------------+
            | Enum Value    | Reference or Description |
            +---------------+--------------------------+
            | owner         | Section 4.4.5            |
            |               |                          |
            | attendee      | Section 4.4.5            |
            |               |                          |
            | optional      | Section 4.4.5            |
            |               |                          |
            | informational | Section 4.4.5            |
            |               |                          |
            | chair         | Section 4.4.5            |
            +---------------+--------------------------+
```

                              Table 12

   Enum Values for scheduleAgent (Context: Participant)

                    +------------+--------------------------+
                    | Enum Value | Reference or Description |
                    +------------+--------------------------+
                    | server | Section 4.4.5 |
                    | | |
                    | client | Section 4.4.5 |
                    | | |
                    | none | Section 4.4.5 |
                    +------------+--------------------------+

                               Table 13

   Enum Values for status (Context: JSEvent)

                    +------------+--------------------------+
                    | Enum Value | Reference or Description |
                    +------------+--------------------------+
                    | confirmed | Section 5.1.3 |
                    | | |
                    | cancelled | Section 5.1.3 |
                    | | |
                    | tentative | Section 5.1.3 |
                    +------------+--------------------------+

                               Table 14

9. Acknowledgments

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

10. Informative References

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

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

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

   [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>.

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

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

   [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>.

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

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

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

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

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