Support for iCalendar Relationships
draft-ietf-calext-ical-relations-08

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 16, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

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

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

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

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

1.1. Structured iCalendar relationships

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

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

1.2. Grouped iCalendar relationships

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

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

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

1.3. Concept relationships

The name CONCEPT is used by the Simple Knowledge Organization System defined in [W3C.REC-skos-reference-20090818]. The term "concept" more accurately defines what we often mean by a category. It’s not the text string that is important but the meaning attached to it. For example, the term "football" can mean very different sports.

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

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

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

1.4. Linked relationships

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

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

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

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

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

Beyond the need to relate elements temporally, project management tools often need to be able to specify the relationships between the various events and tasks which make up a project. The LINK property provides such a mechanism.
The LINK property MUST NOT be treated as just another attachment. The ATTACH property defined in [RFC5545] is being extended to handle server-side management and stripping of inline data. Clients may choose to handle attachments differently from the LINK property as they are often an integral part of the message – for example, the agenda.

For more information on managed attachments see [RFC8607]

1.5. Caching and offline use

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

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

1.6. Conventions Used in This Document

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

2. Reference Types

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

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

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

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

3. Link Relation Types

[RFC8288] defines two forms of relation type: registered and extension. Registered relation types are added to the Link Relations registry as specified in Section 2.1.1 of [RFC8288]. Extension relation types, defined in Section 2.1.2 of [RFC8288], are specified as unique URIs that are not registered in the registry.
The relation types defined in Section 6.1 will be registered with IANA in accordance with the specifications in [RFC8288].

4. New temporal RELTYPE Parameter values

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

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

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

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

\[
\begin{array}{c}
\text{---}
\end{array}
\begin{array}{c}
\text{Task-A}
\end{array}
\begin{array}{c}
\text{---}
\end{array}
\begin{array}{c}
\text{V}
\end{array}
\begin{array}{c}
\text{---}
\end{array}
\begin{array}{c}
\text{Task-B}
\end{array}
\begin{array}{c}
\text{---}
\end{array}
\]

Figure 1: Finish to start relationship

RELTYPE=FINISHTOFINISH: Task-B can only be completed after Task-A is finished. The related tasks may run in parallel before completion.

For example, if the goal is to prepare a meal, we start the potatoes, then the meat then the peas but they should all be cooked at the same time.
RELTYPE=STARTTOFINISH: The start of Task-A (which occurs after Task-B) controls the finish of Task-B. For example, ticket sales (Task-B) end when the game starts (Task-A).

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

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

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

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

6. New Property Parameters

6.1. Link Relation

Parameter name: LINKREL

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

Format Definition:

This parameter is defined by the following notation:

\[
\text{linkreParam} = \text{"REL" ="}
\]

\[
\text{("SOURCE" ; Link to source of this component / DQUOTE uri DQUOTE / iana-token) ; Other IANA registered type}
\]

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

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

Registration: These relation types are registered in [RFC8288]

6.2. Gap

Parameter name: GAP

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

This parameter is defined by the following notation:

\[
gapparam = \text{"GAP"} \text{ "="} \text{dur-value}
\]

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

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

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

7. New Value Data Types

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

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

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

8. New Properties

8.1. Concept

Property name: CONCEPT

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

Value type: URI

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

Conformance: This property can be specified zero or more times in any iCalendar component.
Description: This property is used to specify formal categories or classifications of the calendar component. The values are useful in searching for a calendar component of a particular type and category.

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

This categorization is distinct from the more informal "tagging" of components provided by the existing CATEGORIES property. It is expected that the value of the CONCEPT property will reference an external resource which provides information about the categorization.

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

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

Format Definition:

This property is defined by the following notation:

```
concept  =  "CONCEPT" conceptparam ":" uri CRLF
conceptparam = *(";" other-param)
```

Example:

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

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

8.2. Link

Property name: LINK

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

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

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

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

Format Definition:

This property is defined by the following notation:

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

linkparam = ; the elements herein may appear in any order,
; and the order is not significant.
(";" "VALUE" "=" ("UID" / 
  "URI" / 
  "TEXT"))
1*(";" linkrelparam)
(";" fmttypeparam)
(";" labelparam)
*(";" other-param)
```

Example:

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

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

Example:

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

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

Property name: REFID

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

Value type: TEXT

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

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

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

Format Definition:

This property is defined by the following notation:

```
refid      = "REFID" refidparam "":" text CRLF
  refidparam      = *(";" other-param)
```

Example:

The following is an example of this property.

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

9. Redefined RELATED-TO Property

9.1. RELATED-TO

Property name: RELATED-TO

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

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

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

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

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

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

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

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

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

Format Definition:
This property is defined by the following notation:

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

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

Example:

The following are examples of this property.

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

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

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

10. Security Considerations

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

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

11. IANA Considerations

11.1. iCalendar Property Registrations

The following iCalendar property names have been added to the
iCalendar Properties Registry defined in Section 8.3.2 of [RFC5545]
IANA has also added a reference to this document where the properties
originally defined in [RFC5545] have been updated by this document.
11.2. iCalendar Property Parameter Registrations

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAP</td>
<td>Current</td>
<td>Section 6.2</td>
</tr>
<tr>
<td>REL</td>
<td>Current</td>
<td>Section 6.1</td>
</tr>
</tbody>
</table>

11.3. iCalendar Value Data Type Registrations

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

<table>
<thead>
<tr>
<th>Value Data Type</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFERENCE</td>
<td>Current</td>
<td>Section 7</td>
</tr>
<tr>
<td>UID</td>
<td>Current</td>
<td>Section 7</td>
</tr>
</tbody>
</table>

11.4. iCalendar RELTYPE Value Registrations

The following iCalendar "RELTYPE" values have been added to the iCalendar Relationship Types Registry defined in Section 8.3.8 of [RFC5545]
11.5. New Reference Type Registration

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE</td>
<td>Current</td>
<td>Section 6.1</td>
</tr>
</tbody>
</table>

12. Acknowledgements

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

Adrian Apthorp, Cyrus Daboo, Marten Gajda, Ken Murchison

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

13. References

13.1. Informative References

13.2. Normative References


Author’s Address

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

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

Douglass                 Expires April 16, 2022                [Page 17]
Task Extensions to iCalendar

draft-ietf-calext-ical-tasks-00

Abstract

This document defines extensions to the Internet Calendaring and Scheduling Core Object Specification (iCalendar) (RFC5545) to provide improved status tracking, scheduling and specification of tasks.

It also defines how Calendaring Extensions to WebDAV (CalDAV) (RFC 4791) servers can be extended to support certain automated task management behaviours.

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 23 February 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Acknowledgements

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

John Chaffee, Marten Gajda, Ken Murchison

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

2. Introduction

This document specifies extensions to the existing Internet Calendaring and Scheduling Core Object Specification (iCalendar) [RFC5545], and associated protocols, in order to enhance the structured communication and execution of tasks. The enhancements allow for the communication, time planning and scheduling of tasks by and between automated systems (e.g. in smart power grids, business process management systems) as well as for human centered tasks.

A "task" is a representation of an item of work assigned to an individual or organization. In the iCalendar Object Model [RFC5545] the representation of tasks is by "VTODO" calendar components. Tasks can be identified in a number of situations, either informally as ad-hoc tasks in personal "to-do" lists or more formally in:

* Business processes - ranging from repetitive workflows to adaptive cases and trouble ticketing
* Project Management - whether for large scale construction projects or collaborative software development

The extensions specified here are defined in the context of an overall architecture for task calendaring and scheduling.

2.1. Terms and Definitions

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

Terms defined in this specification include:

Assignee A calendar user assigned to perform a given task. An assignee is equivalent to an attendee of an event.

Calendar User (CU) A person or software system that accesses or modifies calendar information.

Calendar User Agent (CUA) This may be

1. Software with which the calendar user communicates with a calendar service or local calendar store to access calendar information.

2. Software that gathers calendar data on the Calendar User’s behalf.

Candidate A calendar user who might be able to perform a given task, prior to actually being assigned the task, e.g., a dispatcher has a list of taxi drivers (candidates) from which one will be selected to pick-up a passenger.

Organizer A calendar user who creates a calendar item, requests free/busy information, or published free/busy information. It is an Organizer who invites Attendees [RFC5545].

Observer A calendar user interested in a calendar component, e.g., a manager may have interest in all tasks that have not been completed.

Resource A resource in the scheduling context is any shared entity that can be scheduled by a calendar user, but does not control its own attendance status. Resources can be of "Location", "Equipment", or "Role" type.
Task  A representation of an item of work that can be assigned to one or more task actor assignees. In [RFC5545], these are "VTODO" calendar components, which are groupings of component properties and possibly "VALARM" calendar components that represent an action-item or assignment.

3. Task Architecture

A reference architecture for task calendaring and scheduling is defined in order to identify the key logical elements involved in task management and the interfaces between them to enable interoperability. The logical elements identified here establish an appropriate separation of concerns and clarify the responsibilities of different elements. However, the architecture does not prescribe a binding or packaging of elements, i.e., software systems may be developed where some elements are tightly bound and the interfaces between bound elements are not exposed. The task architecture is also described in [TARCH].
4. Task Architecture Elements

The following logical elements form the task architecture that this specification is based on:

Task Actors  Various calendar users that may be involved in the monitoring or performing of a task. The set of actors includes: Organizers, Observers, Resources, Assignees, and Candidates.

Task Organizer  The Organizer of a task.

Task Domain Data  This is any domain specific data that may be acted on or provides context to it in performing a task.

Task Specific Application  A task specific application renders the data concerning the task (including task domain data) for presentation and manipulation by a task actor.

Process Logic  Determines under what conditions a task (or tasks) is generated and the actions to take on completion, or some other status event occurring (or not) on the task.

Task Trigger  This is some event that gives rise to the generation of a task according to Process Logic. Task triggers can come from many different sources including, for example; a task being requested through the calendaring system, a status change in the progression of a business process being managed by a business process management or ERP system.

Task Assignment Rules  Govern how actors are assigned to a task. A range of different assignment patterns [WfRP] may be considered, including the two general cases:

1. Delegation to a named actor or group of actors
2. Advertising to a pool of actors for self-selection

In either case the assignment may be made based on a variety of criteria including, name, availability, skills, capacity, etc.

Task Generating System  A system that creates and assigns tasks in
response to some initiating event (task trigger). Task creation is according to Process Logic with task assignment determined by Task Assignment Rules. This system also tracks the status of tasks and will initiate further actions based upon the status. A task generating system can take many forms, for example; Business Process Management System, Project Management System, Bug Tracking System, Building Control System. A Task Generating System may also be a human. In iCalendar terms the Task Generating System is the organizer.

Human Task Generation Task creation, assignment and tracking coordinated by a human organizer is a special case of a task generating system. In this case Task Assignment Rules and Process Logic may be either explicit or tacit.

Directory Service A software system that stores and provides access to information providing details of task actors that may participate or be interested in a task.

Calendar and Scheduling System A software system that stores, publishes and synchronizes calendar data such as events, tasks and journal entries for actors. In the context of tasks this includes schedules (i.e. allocated time and availability to perform tasks) and task lists. A calendar and scheduling system typically consists of server and client software components.

It is not within the scope of this document to specify how Process Logic or Task Assignment Rules are codified. Such logic and rules may be codified in a variety of ways, including traditional programming languages (e.g. C++, Java) or process modelling languages (e.g. BPMN [BPMN]).

5. Architecture Foundations

The key standards that enable interoperability between the logical elements of the architecture are the Internet Calendaring and Scheduling Core Object Specification (iCalendar) [RFC5545] and associated protocols. Task and task status are represented by the iCalendar "VTODO" component. Protocols include, in particular, the iCalendar Transport-Independent Interoperability Protocol (iTIP) [RFC5546] for task assignment and scheduling, and Calendaring Extensions to WebDAV (CalDAV) [RFC4791] for client server communication.
Additionally, this specification uses definitions from Support for iCalendar Relationships [I-D.ietf-calext-ical-relations]. The LINK, REFID, RELATED-TO and CONCEPT properties enable context and a rich set of relationships between tasks and other iCalendar components to be specified.

6. Task Extensions

In order to support the task architecture described in Section 3, this document defines a number of extensions to the current iCalendar standards in the areas of:

- **Task Specification** improved ability to specify domain specific tasks
- **Task Deadlines, Milestones and Time Planning** clarification of deadlines and extension for task duration to support task time planning
- **Task Scheduling and Assignment** ensure support for common patterns of scheduling and assigning tasks
- **Task Status Tracking** improved granularity in status tracking information and alerting task actors to pending or actual task status changes

These extensions are supported mainly by additions to the properties and parameters used within the "VTODO" component.

7. Task Specification

The specification of tasks must be semantically explicit in order for them to be managed within the context of a business process or project, and be understood by both humans and IT systems. The current VTODO component only provides for simple ad-hoc tasks or 'to do' lists, and is therefore extended by this specification as follows:

- **Task type** explicitly what type of task is to be performed is identified.
- **Task context and relationships** how a specific task relates to other tasks and other objects that need to be understood for the effective execution of a task.
- **Task specific data** the form and content of domain data provided as input to a task and/or that may be output from a task.
- **Organizer and attendee** recognizes that a task organizer or attendee...
can be an automated system.

7.1. CONCEPT for task type identification

The CONCEPT property is used to identify the type of task, for example;

CONCEPT:http://example.com/task/delivery

7.2. Task Context and Relationships

The LINK property specifies a link to external information, which may be context to the task. For example:

LINK;REL=SOURCE:http://example.com/package/1234567890

LINK;REL=describedby:mid:752142.1414823874.307E5@mx123.example.com

The external information may be data to be manipulated in performing the task. See section 3.1.3 Task Domain Data Handling.

REFID is used to identify a key allowing the association of tasks that are related to the same object and retrieval of a task based on this key. This may be, for example, to identify the tasks associated with a given project without having to communicate the task structure of the project, or all tasks associated to a specific package.

REFID:Manhattan

REFID:1234567890

Extensions [Doug114] to the RELATED-TO property allow temporal relationships between tasks as found in project management to be specified as well as parent / child relationships and dependencies (DEPENDS-ON). Tasks (VTODOs) may also be related to other calendar components; for example to a VEVENT to block time to perform a task.

7.3. Task Domain Data Handling

Provide support for task specific input and output data (including updates) beyond the standard iCalendar properties. It is envisaged that standard calendar user agents will be able to launch task specific applications by passing task specific data.

The LINK property can be used to ‘attach’ the domain specific data to the task. For example, it might be a URI pointing to a web page where the status of the task can be directly manipulated.
Or it might be used for attachments specific to the task, for example an electronic copy of a signature taken to confirm delivery of a package.

8. Task Deadlines, Milestones and Time Planning

Deadlines for starting and finishing a task are defined by the DTSTART, DUE and DURATION properties. DTSTART represents the earliest start time for beginning work on a task. DUE, or DTSTART + DURATION represent the latest finish time for a task. Thus these properties define a "window" within which a task has to be performed. However, there is currently no way to indicate how long the task is expected to take. This document defines a new property, ESTIMATED-DURATION, to allow the estimated time that a task should take to be specified separately from the deadlines for starting and finishing a task. This supports time planning by enabling calendar user agents to display when tasks should occur and therefore allow calendar users to visualize when tasks should be performed and allocate time to them.

A task that has intermediary deadlines (i.e., milestones) SHOULD be expressed by child VTODO components (i.e., sub-tasks associated with each of the milestones) in conjunction with the RELATED-TO property to relate the parent and child tasks.

9. Task Scheduling and Assignment

This specification supports the two distinct models of assigning actors to tasks, i.e., 1) strictly one assignee per task or 2) task assignment to multiple assignees. In this regard one or many ATTENDEES may be specified against a task depending upon the model applied by the task organizer.

In addition a number of different patterns of resource or assignee identification are anticipated. The specific Task Assignment Rules are the responsibility of the Task Organizer.

Communication of task assignment or delegation to one or more actors who are allocated to a task by the organizer is directly supported by iTIP, i.e., all included ATTENDEES in an iTIP REQUEST are expected to perform the task.
The offering or advertising of a task to one or more (potential) actors where only one or a subset of the candidates may accept the task will be addressed by a new VPOLL mode (See Appendix B) [VPOLL].

10.  Status Reporting

10.1.  Improved granularity in status reporting information

This document defines new status parameters that can be applied to the VTODO status (STATUS) property, as well as the participant status (PARTSTAT) parameter. These new parameters provide additional information on why (REASON) and when (MODIFIED) a status has changed. In addition to these parameters new status values are specified to provide for task suspension, failure and preparation.

10.2.  Relating comments to status

The GROUP parameter is used with the STATUS or ATTENDEE properties to relate an associated COMMENT property. The COMMENT property can then be used to include additional human readable information about why the associated STATUS or ATTENDEE property changed.

```
STATUS;REASON="http://example.com/reason/delivery-failed"
  SUBSTATE=ERROR;MODIFIED=20130212T120000Z;GROUP=G1:FAILED
COMMENT;MODIFIED=20130226T110451Z;GROUP=G1:Breakdown
ATTENDEE;PARTSTAT=FAILED;MODIFIED=20130226T1104510Z;GROUP=G2:
  REASON="http://example.com/reason/van-break-down"
    mailto:xxx@example.com
COMMENT;MODIFIED=20130226T110451Z;GROUP=G2:Puncture
```

10.3.  Comments associated to reasons and status changes

Reasons may be associated directly with a comment, allowing for multiple reasons associated with a status to each have a comment associated with them [EDISTS].

```
CONCEPT:http://example.com/task/delivery
STATUS;SUBSTATE=ERROR;MODIFIED=20130212T120000Z;GROUP=G1:FAILED
COMMENT;MODIFIED=20130226T110451Z;GROUP=G1:Out of time
COMMENT;REASON="http://example.com/reason/traffic"
  MODIFIED=20130226T110451Z;GROUP=G1:Traffic Accident on E44
COMMENT;REASON="http://example.com/reason/closed"
  MODIFIED=20130226T110451Z;GROUP=G1:Arrived after office hours
```

10.4.  Task Alerts and Notifications

Different needs to alert or notify task actors of pending or actual task status changes are recognized:
Alarms Alarms (VLARM components) operate in the calendar user agent space to notify the task actor of a pending task state for a task they are assigned to or are interested in. Note: there is no constraint in the current standards on the propagation of alarms specified on calendar objects by organizers to individual attendees.

Escalations An escalation or notification to the ATTENDEE, ORGANIZER, or other task actor may be required if a deadline associated with a task is exceeded or for some other reason. Process Logic identifying when and who to propagate escalations to is the responsibility of the Task Generating System, e.g., a BPMS.

Notifications Task actors (observers) not directly involved in performing a task but with a known interest in a given task’s status can be identified by the ASSOCIATE property [Doug214] against certain components e.g. ALARM, to identify which task events the stakeholder/party is interested in. Notifications on shared calendars will allow task actors to register an interest in changes to tasks within a calendar (see Appendix A).

10.5. Automated Status Changes

A new property, TASK-MODE, is introduced to instruct servers to apply automated operations for changing the status of a task.

11. New Property Parameters

11.1. Group

Parameter name GROUP

Purpose This parameter allows the association of different (usually) multiply occurring properties.

Format Definition This parameter is defined by the following notation:

```
groupparam       = "GROUP" "=" text
                 *("," text)
```

Description The value of this parameter is free-form text that creates an identifier for associated properties. All properties that use the same GROUP value are associated through that value. For example, multiple comments and an attendee may be associated with a status value.

Example The following is an example of this property.
11.2. Reason

Parameter name REASON

Purpose To indicate the reason for a change in status of a task or attendee participation status.

Format Definition This parameter is defined by the following notation:

reasonparam = "REASON" "=" DQUOTE uri DQUOTE *("," DQUOTE uri DQUOTE)

Description This property parameter allows the change in status of a task or participant status to be qualified by the reason for the change with a codified reason. Typically reasons are defined within the context of the task type and therefore SHOULD include the namespace of the authority defining the task. Common reason codes are IANA registered and do not have a namespace prefix.

Example

STATUS;REASON="http://example.com/reason/delivered-on-time";
MODIFIED=20130212T120000Z;GROUP=G1:COMPLETED
ATTENDEE;REASON="x-example-reason:out-of-office";
PARTSTAT=DECLINED;MODIFIED=20130212T120000Z;
GROUP=123:mailto:cyrus@example.com

11.3. Modified

Parameter name MODIFIED

Purpose To specify the time and date of when the status of a task or attendee participant status changed.

Format Definition This parameter is defined by the following notation:

modifiedparam = "MODIFIED" "=" date-time

Description The modified parameter allows the specification of the date time of when a status (STATUS) or participant status (PARTSTAT) changed. It MUST be specified in the UTC time format. The value of MODIFIED SHOULD be set at the time when the associated status (either STATUS or PARTSTAT) is changed. Therefore either a client or server may set the value of MODIFIED
depending on which is updating the value of STATUS or PARTSTAT. For backwards compatibility if the server detects that MODIFIED should have changed but wasn’t (for example the client doesn’t support MODIFIED) then the server MAY set MODIFIED retrospectively.

Example

```plaintext
STATUS;REASON="http://example.com/reason/delivered-on-time";
MODIFIED=20130212T120000Z;GROUP=G1:COMPLETED
```

11.4. Sub-State

Parameter name SUBSTATE

Purpose To provide additional granularity of task status for e.g. IN-PROCESS.

Format Definition This parameter is defined by the following notation:

```plaintext
substateparam = "SUBSTATE" "="
               ( "OK" ; everything is fine (the default)
               / "ERROR" ; something is wrong (the REASON code explains why)
               / "SUSPENDED" ; waiting on some other task to complete or availability of a resource (REASON code explains why)
               / x-name ; Experimental type
               / iana-token) ; Other IANA-registered type
```

Description The sub-state parameter allows additional qualification and granularity of states to be recorded, in particular for the IN-PROCESS state. It allows individual sub-states to be recorded without the need to define and publish a sub-task associated with a parent task purely to track that a particular state has been reached. This property also allows parallel states to be expressed e.g. that a task has been suspended at whatever state it has reached.

Example

```plaintext
STATUS;REASON="http://example.com/reason/no-one-home";
SUBSTATE=ERROR:FAILED
STATUS;REASON="http://example.com/reason/paint-drying";
SUBSTATE=SUSPENDED:IN-PROCESS
```
12. New Parameter Values

12.1. Redefined VTODO Participant Status

Participant status parameter type values are defined in [RFC5545].
This specification redefines that type to include the new value
FAILED for VTODO iCalendar components.

Format Definition This property parameter is extended by the
following notation:

```
partstat-todo /= *("FAILED") ; To-do cannot be completed
```

Example

```
ATTENDEE;REASON="http://example.com/reason/not-enough-time";
PARTSTAT=FAILED:mailto:jsmith@example.com
```

13. New Properties

13.1. Estimated Duration

Property Name ESTIMATED-DURATION

Purpose This property specifies the estimated positive duration of
time the corresponding task will take to complete.

Value Type DURATION

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

Conformance This property can be specified in "VTODO" calendar
components.

Format Definition This property is defined by the following
notation:

```
est-duration = "ESTIMATED-DURATION" durparam ":" dur-value CRLF
; consisting of a positive duration of time.
durparam = *(";" other-param)
```

Description In a "VTODO" calendar component the property MAY be used
to specify the estimated duration for the to-do, with or without an explicit time window in which the event should be started and completed. When present, DTSTART and DUE/DURATION represent the window in which the task can be performed. ESTIMATED-DURATION SHOULD be passed from ORGANIZER to ATTENDEE in iTIP [RFC5546] messages.

Example The following is an example of this property that specifies an interval of time of exactly one hour:

ESTIMATED-DURATION:PT1H

13.2. Task Mode

Property Name TASK-MODE

Purpose This property specifies automatic operations that servers apply to tasks based on changes in attendee status (PARTSTAT).

Value Type TEXT

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

Conformance This property can be specified zero or more times in a "VTODO" calendar component.

Format Definition This property is defined by the following notation:

\[
\text{task-mode} = \text{"TASK-MODE taskmodeparam ":" taskvalue } \\
\text{""," taskvalue) CRLF}
\]

\[
\text{taskvalue} = \text{"AUTOMATIC-COMPLETION" ; set STATUS completed if all attendees have completed} \\
\text{"AUTOMATIC-FAILURE"} \\
\text{"SERVER"} \\
\text{"CLIENT"} \\
\text{iana-token} \\
x-name
\]

\[
\text{taskmodeparam = ""," other-param}
\]

Description In a "VTODO" calendar component this property MAY be used to indicate to servers how they can automatically change the state of the task based on iTIP replies from Attendees. For example, the server can automatically set the overall task status (STATUS) to COMPLETED when every attendee has marked their own
status (PARTSTAT) as COMPLETED, or the server could mark the task as FAILED if its DUE date passes without it being completed. TASK-MODE processing is performed on the organizer’s copy of the task.

The property value is a list of one or more IANA registered tokens that defines modes to be used for the task. This specification defines three modes which are described in the following sub-sections.

Examples

TASK-MODE: AUTOMATIC-COMPLETION, AUTOMATIC-FAILURE
TASK-MODE: SERVER
TASK-MODE: AUTOMATIC-FAILURE

AUTOMATIC-COMPLETION Task Mode The task mode value "AUTOMATIC-COMPLETION" indicates to the server that it can change the "VTODO" component’s STATUS property value to "COMPLETED" as soon as all ATTENDEEs in the task have replied with a "PARTSTAT" parameter set to "COMPLETED".

AUTOMATIC-FAILURE Task Mode The task mode value "AUTOMATIC-FAILURE" indicates to the server that it SHOULD change the "VTODO" component’s STATUS property value to "FAILED" if either:

* the PARTSTAT of one ATTENDEE is set to FAILED; or

* the current time is past the effective due date of the component and the task has not yet been completed.

Note: The effective due date is either the "DUE" property value or the combination of the "DTSTART" and "DURATION" property values.

CLIENT Task Mode The task mode value "CLIENT" is an instruction to the server to honour the status set by the client.

SERVER Task Mode The task mode value "SERVER" indicates to the server that it can change the "VTODO" component’s STATUS property value to an appropriate value, based on implementation defined "business rules", as ATTENDEE responses are processed or as deadlines related to the task pass.

The server can add this property to a "VTODO" component to indicate to the client that it will be managing the status.

14. Property Extensions and Clarifications
14.1. The ATTENDEE property

The Attendee property is defined in [RFC5545]. This specification extends that property to include new parameters to indicate the reason for a participant status change (See Appendix A) and sub-states.

Format Definition  This property is defined by the following notation:

```plaintext
attendee   = "ATTENDEE" attparam ";" cal-address CRLF
attparam   /= *(
    ; The following are OPTIONAL, but MUST NOT occur more than once.
    ;
    (";" reasonparam)
    (";" modifiedparam)
    (";" substateparam)
)
```

Example: The following are examples of this property’s use for tasks:

```plaintext
ATTENDEE;PARTSTAT=DECLINED;MODIFIED=20130212T120000Z;GROUP=G1;
   REASON="http://example.com/reason/too-busy":mailto:xxx@example.com

ATTENDEE;PARTSTAT=IN-PROCESS;MODIFIED=20130212T120000Z;
   SUBSTATE=X-EXAMPLE-STEP-1:mailto:xxx@example.com
```

14.2. Redefined COMMENT Property Parameter List

The Comment property is defined in [RFC5545].

Format Definition  The "COMMENT" property parameter list is augmented as follows:

```
commparam /= *
    ; The following are OPTIONAL, but MUST NOT occur more than once.
    (";" reasonparam) /
    (";" modifiedparam)
```

Apthorp & Douglass Expires 23 February 2022 [Page 19]
14.3. Redefined STATUS Property

The Status property is defined in [RFC5545]. This specification extends that property to include new parameters to indicate the reason for a status change as well as new values associated with VTODO iCalendar components (See Appendix A for examples of the task state lifecycle).

Format Definition The "STATUS" property parameter list is augmented as follows:

```
statparam /= *( ; The following are OPTIONAL,
; but MUST NOT occur more than once.
; (";" reasonparam)
(";" modifiedparam)
(";" substateparam) / )
```

```
statvalue-todo = / "PENDING" ;Indicates a to-do has been
;created and accepted, but has not
;yet started.
/ "FAILED" ;Indicates to-do has failed.
;Extended status values for
;"VTODO".
```

Description:

PENDING - A task has been created but has not yet started and is ready to start subject to other dependencies (e.g. preceding task or DTSTART). This is the default state.

FAILED - task has failed and may need some follow-up from the organizer to re-schedule or cancel

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

```
STATUS;REASON="http://example.com/reason/delivery-failed";
SUBSTATE=ERROR;MODIFIED=20130212T120000Z;GROUP=G1:FAILED
```
15.  CalDAV Support for Task Mode

The CalDAV [RFC4791] calendar access protocol allows clients and servers to exchange iCalendar data. With the introduction of the "TASK-MODE" property in this specification, different automated task management behaviours may be delegated to the server by the Task Organizer depending upon the value of "TASK-MODE".

In order for a CalDAV client to know what task modes are available, a CalDAV server advertises a CALDAV:supported-task-mode-set WebDAV property on calendar home or calendar collections if it supports the use of the "TASK-MODE" property as described in this specification. The server can advertise a specific set of supported task modes by including one or more CALDAV:supported-task-mode XML elements within the CALDAV:supported-task-mode-set XML element. If no CALDAV:supported-task-mode XML elements are included in the WebDAV property, then clients can try any task mode, but need to be prepared for a failure when attempting to store the calendar data.

Clients MUST NOT attempt to store iCalendar data containing "TASK-MODE" elements if the CALDAV:supported-task-mode-set WebDAV property is not advertised by the server.

The server SHOULD return an HTTP 403 response with a DAV:error element containing a CALDAV:supported-task-mode XML element, if a client attempts to store iCalendar data with an "TASK-MODE" element value not supported by the server.

It is possible for a "TASK-MODE" value to be present in calendar data on the server being accessed by a client that does not support the "TASK-MODE" property. It is expected that existing clients, unaware of "TASK-MODE", will fail gracefully by ignoring the calendar property.

15.1.  CALDAV:supported-task-mode-set Property

Name   supported-task-mode-set

Namespace  urn:ietf:params:xml:ns:caldav

Purpose  Enumerates the set of supported iCalendar "TASK-MODE" element values supported by the server.

Protected  This property MUST be protected and SHOULD NOT be returned by a PROPFIND allprop request (as defined in Section 14.2 of [RFC4918]).

Description  See above.
Definition

<!ELEMENT supported-task-mode-set(supported-task-mode*)>
<!ELEMENT supported-task-mode (#PCDATA)>
<!-- PCDATA value: string - case insensitive but uppercase preferred -->

Example

<C:supported-task-mode-set xmlns:C="urn:ietf:params:xml:ns:caldav">
  <C:supported-task-mode>AUTOMATIC-COMPLETION</C:supported-task-mode>
  <C:supported-task-mode>AUTOMATIC-FAILURE</C:supported-task-mode>
  <C:supported-task-mode>SERVER</C:supported-task-mode>
  <C:supported-task-mode>CLIENT</C:supported-task-mode>
</C:supported-task-mode-set>

16. Security Considerations

This specification introduces no new security considerations beyond those identified in [RFC5545].

17. IANA Considerations

17.1. Initialization of the Status registry

This specification updates [RFC5545] by adding a Status value registry to the iCalendar Elements registry and initializing it as per [RFC5545].
Table 1: Initial Status Value Registry

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCELLED</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>COMPLETED</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>CONFIRMED</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>DRAFT</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>FINAL</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
<tr>
<td>TENTATIVE</td>
<td>Current</td>
<td>[RFC5545]</td>
</tr>
</tbody>
</table>

17.2. Update of the Status registry

This specification further updates the Status registry with additional values defined in this document.

<table>
<thead>
<tr>
<th>Value</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PENDING</td>
<td>Current</td>
<td>This Spec, Section 14.3</td>
</tr>
<tr>
<td>FAILED</td>
<td>Current</td>
<td>This Spec, Section 14.3</td>
</tr>
</tbody>
</table>

Table 2: Updated Status Value Registry

17.3. Sub-State value registry

The following table has been used to initialize the Sub-State registry.
17.4. Task Mode value registry

The following table has been used to initialize the Task Mode registry.

<table>
<thead>
<tr>
<th>Task Mode</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMATIC-COMPLETION</td>
<td>Current</td>
<td>This Spec, Section 13.2</td>
</tr>
<tr>
<td>AUTOMATIC-FAILURE</td>
<td>Current</td>
<td>This Spec, Section 13.2</td>
</tr>
<tr>
<td>CLIENT</td>
<td>Current</td>
<td>This Spec, Section 13.2</td>
</tr>
<tr>
<td>SERVER</td>
<td>Current</td>
<td>This Spec, Section 13.2</td>
</tr>
</tbody>
</table>

Table 4: Task Mode Value Registry

17.5. Participation Statuses registry

The following table has been used to update the Participation Statuses registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED</td>
<td>Current</td>
<td>This Spec, Section 12.1</td>
</tr>
</tbody>
</table>

Table 5: Participation Statuses Registry

17.6. Properties registry

The following table has been used to update the Properties registry.
### Table 6: Updated Properties Registry

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTENDEE</td>
<td>Current</td>
<td>This Spec, Section 14.1</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Current</td>
<td>This Spec, Section 14.2</td>
</tr>
<tr>
<td>ESTIMATED_DURATION</td>
<td>Current</td>
<td>This Spec, Section 13.1</td>
</tr>
<tr>
<td>STATUS</td>
<td>Current</td>
<td>This Spec, Section 14.3</td>
</tr>
<tr>
<td>TASK-MODE</td>
<td>Current</td>
<td>This Spec, Section 13.2</td>
</tr>
</tbody>
</table>

17.7. Parameters registry

The following table has been used to update the Parameters registry.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REASON</td>
<td>Current</td>
<td>This Spec, Section 11.2</td>
</tr>
<tr>
<td>MODIFIED</td>
<td>Current</td>
<td>This Spec, Section 11.3</td>
</tr>
<tr>
<td>SUBSTATE</td>
<td>Current</td>
<td>This Spec, Section 11.4</td>
</tr>
</tbody>
</table>

### Table 7: Updated Parameters Registry

18. Normative References


[I-D.ietf-calext-eventpub-extensions]

[I-D.ietf-calext-ical-relations]

19. Informative References


[I-D.york-vpoll]

[TARCH]


Appendix A. Examples of Task State Lifecycle

A.1. Simple Case Status Change

<table>
<thead>
<tr>
<th></th>
<th>STATUS</th>
<th>PARTSTAT</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Organizer draft</td>
</tr>
<tr>
<td>2</td>
<td>NEEDS-ACTION</td>
<td>NEEDS-ACTION</td>
<td>Organizer sends iTIP request</td>
</tr>
<tr>
<td>3</td>
<td>NEEDS-ACTION</td>
<td>ACCEPTED</td>
<td>Attendee reply</td>
</tr>
<tr>
<td>4</td>
<td>PENDING</td>
<td>ACCEPTED</td>
<td>Task accepted but waiting on some &quot;trigger&quot; to start (e.g. another task has to finish first)</td>
</tr>
<tr>
<td>5</td>
<td>IN-PROCESS</td>
<td>IN-PROCESS</td>
<td>Attendee reply now working on the task</td>
</tr>
<tr>
<td>6</td>
<td>IN-PROCESS</td>
<td>COMPLETED</td>
<td>Attendee reply completed</td>
</tr>
<tr>
<td>7</td>
<td>COMPLETED</td>
<td>COMPLETED</td>
<td>Organizer changes overall state</td>
</tr>
</tbody>
</table>

Table 8: Example of status changes in assigning and performing a task with one attendee.

A.2. Example for multiple Attendees

Example of status changes in assigning and performing a task with two attendees (A1 and A2).
<table>
<thead>
<tr>
<th>STATUS</th>
<th>PARTSTAT (A1)</th>
<th>PARTSTAT (A2)</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Organizer draft.</td>
</tr>
<tr>
<td>2</td>
<td>NEEDS-ACTION</td>
<td>NEEDS-ACTION</td>
<td>Organizer sends iTIP request.</td>
</tr>
<tr>
<td>4</td>
<td>NEEDS-ACTION</td>
<td>ACCEPTED</td>
<td>Attendee 1 reply.</td>
</tr>
<tr>
<td>5</td>
<td>NEEDS-ACTION</td>
<td>ACCEPTED</td>
<td>Attendee 2 reply.</td>
</tr>
<tr>
<td>6</td>
<td>PENDING</td>
<td>ACCEPTED</td>
<td>Task accepted but waiting on some &quot;trigger&quot; to start (e.g. another task has to finish first)</td>
</tr>
<tr>
<td>7</td>
<td>IN-PROCESS</td>
<td>ACCEPTED</td>
<td>Attendee 2 reply now working on the task.</td>
</tr>
<tr>
<td>8</td>
<td>IN-PROCESS</td>
<td>IN-PROCESS</td>
<td>Attendee 1 reply now working on the task.</td>
</tr>
<tr>
<td>9</td>
<td>IN-PROCESS</td>
<td>COMPLETED</td>
<td>Attendee 1 reply Completed (overall status still IN-PROCESS).</td>
</tr>
<tr>
<td>10</td>
<td>IN-PROCESS</td>
<td>COMPLETED</td>
<td>Attendee 2 reply Completed</td>
</tr>
<tr>
<td>11</td>
<td>COMPLETED</td>
<td>COMPLETED</td>
<td>Organizer changes</td>
</tr>
</tbody>
</table>
Table 9: Example for multiple Attendees

<table>
<thead>
<tr>
<th>STATUS</th>
<th>PARTSTAT</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>Organizer draft</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>NEEDS-ACTION</td>
<td>Organizer sends iTIP request</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>ACCEPTED</td>
<td>Attendee reply</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>IN-PROCESS</td>
<td>Attendee reply now working on the task</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>FAILED</td>
<td>Attendee reply task failed</td>
</tr>
<tr>
<td>FAILED</td>
<td>FAILED</td>
<td>Organizer changes overall state</td>
</tr>
</tbody>
</table>

Table 10: Example of Failure

A.3. Example of Failure

Example of status changes for a task that fails.

Appendix B. Change log

V02. 2021-05-05 MD

* Redo in asciidoc

* Change STRUCTURED-CATEGORY to CONCEPT

* Add GROUP parameter definition

V01. 2015-08-23 AA
* Highlighted use of ESTIMATED-DURATION for time planning.

* Corrected PARTSTAT example section 5.1. Changed DECLINED to FAILED.

* Replaced Task Mode AUTOMATIC-STATUS with CLIENT and SERVER modes. Also, clarified that task mode processing is only done on the organizer’s copy.

* Clarified responsibility for setting MODIFIED.

* CalDAV support added.

* Updated normative references.

Appendix C. Working Notes

C.1. Advertising tasks

Use VPOLL for advertising a task to a pool of possible ATTENDEEs and then select the respondent to assign one or more assignees.

Introduce POLL-MODE:ASSIGNMENT

Need to indicate number of assignees required.

Potentially different types of response e.g. ACCEPT or DECLINE, or a weighting e.g. 0 – 100

Take into FREEBUSY discussion.

C.2. Subscribing to task updates

Stakeholders should have the ability to subscribe to categories / types of tasks on an ongoing basis. Reference calendarserver.org notifications draft

Authors’ Addresses

Adrian Apthorp
DHL Express

Email: aapthorp@theiet.org

Michael Douglass
Bedework Commercial Services
Abstract

This document provides the required methods for converting JSCalendar from and to iCalendar.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 10, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
# Table of Contents

1. Introduction .................................................. 4  
   1.1. Note (to be deleted later) ......................... 4  
   1.2. Motivation ........................................ 4  
   1.3. Scope and caveats ................................ 4  
   1.4. Notational Conventions ............................. 5  
2. iCalendar pre-processing .................................. 5  
3. Translating iCalendar components to JSCalendar .......... 6  
   3.1. VALARM ............................................. 7  
   3.2. VCALENDAR .......................................... 9  
   3.3. VEVENT ............................................. 10  
   3.4. VLOCATION .......................................... 10  
   3.5. VTIMEZONE, STANDARD, DAYLIGHT ................. 11  
   3.6. VTODO ............................................... 12  
4. Translating iCalendar properties to JSCalendar ........ 13  
   4.1. ATTACH ............................................ 13  
   4.2. ATTENDEE ........................................... 13  
   4.3. CALSCALE ........................................ 16  
   4.4. CATEGORIES ......................................... 16  
   4.5. CLASS ............................................... 16  
   4.6. COLOR ................................................ 17  
   4.7. COMMENT ............................................ 17  
   4.8. COMPLETED ......................................... 17  
   4.9. CONCEPT ............................................ 17  
   4.10. CONFERENCE ....................................... 18  
   4.11. CONTACT .......................................... 18  
   4.12. CREATED .......................................... 19  
   4.13. DESCRIPTION ..................................... 19  
   4.14. DTEND, DTSTART, DUE, DURATION .............. 20  
   4.15. ESTIMATED-DURATION .............................. 20  
   4.16. EXDATE ............................................ 23  
   4.17. EXRULE ............................................ 23  
   4.18. DTSTAMP and LAST-MODIFIED ................... 23  
   4.19. GEO ................................................. 24  
   4.20. IMAGE ............................................... 24  
   4.21. LOCATION ......................................... 24  
   4.22. METHOD ............................................ 24  
   4.23. ORGANIZER ......................................... 25  
   4.24. PERCENT-COMPLETE ............................... 25  
   4.25. PRIORITY ......................................... 27  
   4.26. PRODID ............................................ 28  
   4.27. RECURRENCE-ID .................................... 28  
   4.28. RELATED-TO ....................................... 28  
   4.29. REQUEST-STATUS ................................. 29  
   4.30. RESOURCES ........................................ 29  
   4.31. RDATE ............................................. 30  
   4.32. RRULE ............................................. 30
5. Translating iCalendar Recurrences
   5.1. Translating iCalendar Recurrences: Simple objects with overrides
   5.2. Translating iCalendar Recurrences: Overrides with no master
6. Translating iCalendar: Further examples
   6.1. Recurring event with ATTACH
   6.2. Simple event with CONTACT
   6.3. Simple event with RESOURCES
   6.4. Recurring event. Attendees only in overrides
7. Translating JSCalendar properties to iCalendar components
   7.1. jsevent
   7.2. jsgroup
   7.3. jstask
   7.4. timezones
   7.5. alerts
      7.5.1. action
      7.5.2. trigger
      7.5.3. todo
8. Translating JSCalendar properties to iCalendar properties
   8.1. categories
   8.2. created
   8.3. duration
   8.4. estimatedDuration
   8.5. keywords
   8.6. locations
   8.7. participants
9. Security Considerations
10. IANA Considerations
11. Acknowledgments
12. References
   12.1. Normative References
   12.2. Informative References
Appendix A. Outdated document sections
   A.1. Note
   A.2. JSTask
   A.3. JSGroup
   A.4. Common properties
      A.4.1. Time properties and types
      A.4.2. Locations
      A.4.3. Participants
1. Introduction

1.1. Note (to be deleted later)

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

1.2. Motivation

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

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

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

1.3. Scope and caveats

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

- JSCalendar contains a richer data model to express calendar information such as event locations and participants. While future iCalendar extensions may allow a direct mapping, for now there may be no representation directly in iCalendar of some properties. These values may have to be extracted from a full copy of the iCalendar format provided as a property in the JSCalendar data.

- iCalendar may contain arbitrary, non-standardised data with custom properties/attributes. These will be translated using the same approach as jCal.
o iCalendar has some obsolete features that have been removed from JSCalendar due to not being useful and/or supported in the real world (e.g. custom email alerts to send to random people). Translating these may lose some of the original fidelity.

o Implementations may use a custom property to store data that could not be mapped directly in either direction in the original or a custom format, however this is not interoperable.

o JSCalendar supports fractional seconds in time values whereas iCalendar does not. A subsequent specification will define how fractional seconds can be represented in iCalendar.

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

This document defines mappings for the following specifications.

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

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

1.4. Notational Conventions

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

2. iCalendar pre-processing

iCalendar uses a line-folding mechanism to limit lines of data to a maximum line length (typically 75 octets) to ensure the maximum likelihood of preserving data integrity as it is transported via various means (e.g., email) -- see Section 3.1 of [RFC5545].
iCalendar data uses an "escape" character sequence for text values and property parameter values. See Sections 3.1 and 3.3 of [RFC5545] as well as [RFC6868].

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

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

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

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

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

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

3. Translating iCalendar components to JSCalendar

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

At present VFREEBUSY and VJOURNAL are not mapped in jscalendar.
3.1. VALARM

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

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

maps to

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

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

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

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

Specifying that a client should send emails to all attendees is both annoying and dangerous. Attendees have their own preferences for how and when they should be notified.
Accordingly, the specification only allows for "display" and "email" actions and - other than specifying when - does not allow much else. Clients and/or servers will generally use the associated event or task title as identification. User preferences generally indicate what actions they prefer.

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

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

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

BEGIN:VEVENT
... 
BEGIN:VALARM
TRIGGER;VALUE=DATE-TIME:19970317T133000Z
REPEAT:4
DURATION:PT15M
ACTION:DISPLAY
ATTACH;FMTTYPE=audio/basic:ftp://example.com/pub/
sounds/bell-01.aud
END:VALARM
BEGIN:VALARM
TRIGGER:-PT30M
REPEAT:2
DURATION:PT15M
ACTION:DISPLAY
DESCRIPTION:Breakfast meeting with executive\n
  team at 8:30 AM EST.
END:VALARM
BEGIN:VALARM
TRIGGER;RELATED=END:-P2D
ACTION:EMAIL
ATTENDEE:mailto:john_doe@example.com
SUMMARY:*** REMINDER: SEND AGENDA FOR WEEKLY STAFF MEETING ***
DESCRIPTION:A draft agenda needs to be sent out to the attendees to the weekly managers meeting (MGR-LIST). Attached is a pointer the document template for the agenda file.
ATTACH;FMTTYPE=application/msword:http://example.com/
templates/agenda.doc
END:VALARM
END:VEVENT
{  
   "@type": "Event",
   ...
   "alerts": {
      "1": {
         "@type": "Alert",
         "action": "display",
         "trigger": {
            "@type": "AbsoluteTrigger",
            "when": "19970317T133000Z"
         }
      },
      "2": {
         "@type": "Alert",
         "action": "display",
         "trigger": {
            "@type": "OffsetTrigger",
            "offset": "-PT30M"
         }
      },
      "3": {
         "@type": "Alert",
         "action": "email",
         "trigger": {
            "@type": "OffsetTrigger",
            "offset": "-P2D",
            "relativeTo": "end"
         }
      }
   }
}

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

3.2. VCALENDAR

A [RFC5545] VCALENDAR component may be mapped to a JSCalendar object with a type of "jsgroup".
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
...
END: VCALENDAR

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

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

3.3. VEVENT

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

BEGIN: VEVENT
...
END: VEVENT

maps to
{
   "@type": "Event",
   ...
}

3.4. VLOCATION

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

maps to

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

3.5. VTIMEZONE, STANDARD, DAYLIGHT

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

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

Note that

- There is no current approach for defining standalone sets of timezones.
- Timezones defined in the IANA timezone database SHOULD NOT be redefined in the object. Only custom timezones will be defined.
BEGIN: VTIMEZONE
TZID: Example/Somewhere
...
END: VTIMEZONE
BEGIN: VTIMEZONE
TZID: Example/Somewhere-else
...
END: VTIMEZONE
BEGIN: VEVENT
...
END: VEVENT

maps to

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

3.6. VTODO

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

BEGIN: VTODO
...
END: VTODO

maps to

{
  "@type": "jstask",
  ...
}
4. Translating iCalendar properties to JSCalendar

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

4.1. ATTACH

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

- A uri value
- A binary value

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

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

For a binary value use a base64 data uri.

For an example of a recurring event with ATTACH see Section 6.1

4.2. ATTENDEE

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

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

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

For example:
maps to

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

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

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

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

INDIVIDUAL  "individual"

GROUP  "group"

RESOURCE  "resource"

ROOM  "location"

UNKNOWN  No value

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

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

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

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

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

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

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

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

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

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

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

   CHAIR  "attendee" and "chair"
   REQ-PARTICIPANT  "attendee"
   OPT-PARTICIPANT  "attendee" and "optional"
   NON-PARTICIPANT  "informational"

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

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

SCHEDULE-AGENT: If the value is "CLIENT" (ignoring case) set the JSCalendar "scheduleAgent" property to "client" otherwise omit it.
SCHEDULE-FORCE-SEND: Set the JSCalendar "scheduleForceSend" property to the lower-cased value of the [RFC6638] SCHEDULE-FORCE-SEND parameter.

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

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

4.3. CALSCALE

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

4.4. CATEGORIES

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

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

maps to

```
...
"keywords": {
  "APPOINTMENT": true,
  "EDUCATION": true,
  "MEETING": true
},
```

4.5. CLASS

Maps to the "privacy" property. The iCalendar property value maps to the JSCalendar value as follows:

CONFIDENTIAL "secret"

PRIVATE "private"

PUBLIC "public"

iana-token and x-name verbatim copy
4.6. COLOR

Maps to the "color" property. Copy the verbatim value.

4.7. COMMENT

There is no direct mapping for this property which may appear multiple times in [RFC5545].

For a scheduling reply it is presumably a message by the participant so the value or values should be used to set the JSCalendar "participantComment" property.

4.8. COMPLETED

Set the JSCalendar "progress" property to "completed" and the "progressUpdated" property to the reformatted date/time.

... COMPLETED: "2010-10T10:10:10Z"
...
maps to

... "progressUpdated": "2010-10-10T10:10:10Z",
"progress": "completed",
...

4.9. CONCEPT

This [draft-ietf-calext-ical-relations] property may appear multiple times in components.

Each instance of the property is mapped on to a member of the JSCalendar "categories" property.
CONCEPT:http://example.com/event-types/arts/music
CONCEPT:http://example.com/performance-types/arts/live

maps to

"categories": {
    "http://example.com/event-types/arts/music": true,
    "http://example.com/performance-types/arts/live": true
}

4.10. CONFERENCE

Maps to a "VirtualLocation" object. The property value maps to the "uri" property of the virtual location.

Mapping parameters:

FEATURE: Maps to the "features" property of the virtual location.

LABEL: Maps to the "name" property of the virtual location.

LANGUAGE: No mapping.

4.11. CONTACT

The CONTACT property is mapped on to a participant object with a "roles" property of "contact" and an "order" property of 1 (one). This defines the participant as a primary contact.

Mapping parameters:

ALTREP  Use the same process as for the ATTENDEE DIR parameter:
        create a link property with the "rel" property set to "alternate"
        and the "href" property set to the value of the ALTREP parameter.
        Then add the link to the participants "links" property.

LANG  Set the participants "language" property.

For an example see Section 6.2
4.12. CREATED

The CREATED property is mapped on to a "created" property with a json formatted form of the date. Example:

```
BEGIN:VEVENT
...
CREATED:19960329T133000Z
...
END:VEVENT
```

maps to

```
{
   "@type": "jsevent",
   ...
   "created": "1996-03-29T13:30"00Z",
   ...
}
```

4.13. DESCRIPTION

Copy the value, preprocessed according to Section 2 into the "description" property.

Mapping parameters:

ALTREP  No mapping.

LANG  Use the "locale" property.

Example:
BEGIN:VEVENT
...
DESCRIPTION:We are having a meeting all this week at 12 pm for one hour, with an additional meeting on the first day 2 hours long.
Please bring your own lunch for the 12 pm meetings.
...
END:VEVENT

maps to

{
    "@type": "jsevent",
    ...
    "description": // Note: comments and string concatenation are not allowed per the JSON specification and is used here to avoid long lines. 
        "We are having a meeting all this week at 12 pm for one " + 
        "hour, with an additional meeting on the first day 2 " + 
        "hours long.\n        Please bring your own lunch for the 12 pm " + 
        "meetings."

    ...
}

4.14. DTEND, DTSTART, DUE, DURATION

If the DTSTART is a DATE only property then add the JSCalendar showWithoutTime property with the value set to "true". The JSCalendar "start" property is set with zero time values.

If the DTSTART has a TZID parameter then set the JSCalendar "timeZone" property to the value of TZID.

If the DTSTART has a UTC value then set the JSCalendar "timeZone" property to the value "Etc/UTC". The JSCalendar "start" property is set without any UTC indicator.

JSCalendar has no equivalent to DTEND. If the component has a DTEND then calculate a value for "DURATION" from that property and DTSTART and proceed as below.

If the DTEND has a TZID parameter with a value that differs from the DTSTART TZID parameter then a "location" object should be created with a "relativeTo" property set to "end" and a "timezone" property set to the value of the "TZID" parameter.

Note that a task is not required to have a DTSTART so the JSCalendar "timezone" property needs to be set from the DUE property.
Convert a DURATION property to the JSCalendar duration.

Example - DTSTART and DTEND in same timezone:

BEGIN:VEVENT
...
DTSTART;TZID=America/New_York:20170315T150000
DTEND;TZID=America/New_York:20170315T160000
...
END:VEVENT

maps to

{
   "@type": "jsevent",
   ...
   "start": "2017-03-15T15:00:00",
   "timeZone": "America/New_York",
   "duration": "PT1H"
   ...
}

Example - DTSTART and DTEND in different timezone:
BEGIN:VEVENT
... 
DTSTART;TZID=America/New_York:20170315T150000
DTEND;TZID=America/LosAngeles:20170315T190000
...
END:VEVENT

maps to

{  
  "@type": "jsevent",
  ...
  "start": "2017-03-15T15:00:00",
  "timeZone": "America/New_York",
  "duration": "PT7H"
...
  "locations": {  
    "1": {  
      "@type": "location",
      "relatedTo": "end",
      "timeZone": "America/Los_Angeles"
    }
  }
}

Example – 3 day event:

BEGIN:VEVENT
...
DTSTART;VALUE=DATE:20210315
DTEND;VALUE=DATE:20210318
...
END:VEVENT

maps to

{  
  "@type": "jsevent",
  ...
  "start": "2017-03-15T00:00:00",
  "duration": "P3D",
  "showWithoutTime": true,
  ...
}
4.15. ESTIMATED-DURATION

Copy the ESTIMATED-DURATION value into the JSCalendar "estimatedDuration" property.

For example:

...  
ESTIMATED-DURATION:PT18H  
...  

maps to

...  
"estimatedDuration": "PT18H"  
...  

4.16. EXDATE

Create a patch object with the recurrence id set from the EXDATE value. Add a single JSCalendar "excluded" property with the value set to true. There MUST NOT be any other properties set - other than "@type".

4.17. EXRULE

Maps to the "excludedRecurrenceRules" property. Also see Section 4.32.

4.18. DTSTAMP and LAST-MODIFIED

The mapping depends on whether or not the component is a scheduling entity.

Not a scheduling entity: The [RFC5545] DTSTAMP and LAST-MODIFIED properties have essentially the same meaning. If both are present use the value of the latest for the "updated" property. Otherwise set from whichever is present.

Is a scheduling entity: DTSTAMP should be used to set the "ScheduleUpdated" property in the "participant" object for the attendee.

If present LAST-MODIFIED should be used to set the "updated" property - otherwise set it from the DTSTAMP.
4.19. GEO

Maps to a Location object, with only the "coordinates" property set. Note that the JSCalendar coordinates property value MUST be a valid "geo" URI, so replace the ";" character in the iCalendar value with "," and prepend the resulting string with "geo:"

4.20. IMAGE

Maps to a Link object with the iCalendar property value mapped to the location "href" property, and the "rel" property set to "icon". For a binary value use a base64 data uri in the "href" property.

Mapping parameters:
ALTREP  No mapping.
FMTTYPE  Maps to the "contentType" property of the Link object.
DISPLAY  Maps to the "display" property of the Link object. The property values "BADGE", "GRAPHIC", "FULLSIZE" and "THUMBNAIL" map to their lower-case equivalent in JSCalendar.

4.21. LOCATION

If any [draft-ietf-calext-eventpub-extensions] "VLOCATION" components are present, then the [RFC5545]"LOCATION" property should be ignored.

To map the property create a "locations" property with a single "location" and set the "description" property to the value of the [RFC5545]"LOCATION" property.

Mapping parameters:
ALTREP  Maps to a Link object in the Location "links" property, with the "href" property set to the parameter value.

4.22. METHOD

Maps to the "method" property of the JSCalendar object. The JSCalendar property value is the lowercase equivalent of the iCalendar property value.
4.23. ORGANIZER

Maps to the "replyTo" property of the JSCalendar object. An iCalendar property value in the "mailto:" URI scheme, maps to the "imip" method, any other value maps to the "other" method.

If the iCalendar component also contains an ATTENDEE with the same calendar user address then map that ATTENDEE as defined in Section 4.2 and add the "owner" role to the Participant "roles" property. Otherwise, use the ORGANIZER property to map to a Participant object. The "roles" property of the Participant MUST only contain the "owner" role and the "expectReply" property value MUST be "false". Any iCalendar parameters map as defined for ATTENDEE.

TBD: SENT-BY parameter. Example.

4.24. PERCENT-COMPLETE

For all methods other than REPLY (or no method), the PERCENT-COMPLETE applies to the VTODO as a whole. In this case it the value is used to set the JSCalendar "percentComplete" property in the task object.

BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
METHOD:PUBLISH
BEGIN:VTODO
...
PERCENT-COMPLETE:39
END:VTODO
END: VCALENDAR

maps to

{
   "@type": "jstask",
   "prodid": "-//ABC Corporation//NONSGML My Product//EN",
   ...
   "percentComplete": 39
}

PERCENT-COMPLETE in a REPLY is used to indicate the level of completeness of the ATTENDEE. There should only be a single ATTENDEE in the VTODO object.

As ever recurrences complicate matters. For a non-recurring event or an override that contains the single participant, set the JSCalendar...
"percentComplete" property in the JSCalendar "participant" object representing the attendee.

BEGIN:VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
METHOD:REPLY
BEGIN:VTODO

... 
ATTENDEE:mailto:douglm@example.org
PERCENT-COMPLETE:39
END:VTODO
END:VCALENDAR

maps to

{
    "@type": "jstask",
    "prodid": "-//ABC Corporation//NONSGML My Product//EN",
    ...
    "participants": {
        "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
            "@type": "Participant",
            "sendTo": {
                "imip": "mailto:douglm@example.org"
            },
            "percentComplete": 39,
            "roles": {
                "attendee": true
            }
        },
        ...
    }
}

In the case of an override with the participant appearing in the master then add a patch to the override.
BEGIN: VCALENDAR
PRODID:-//ABC Corporation//NONSGML My Product//EN
METHOD:REPLY
BEGIN: VTODO
... 
ATTENDEE: mailto:douglm@example.org
END: VTODO
BEGIN: VTODO
... 
RECURRENCE-ID: 20200523T120000
... 
ATTENDEE: mailto:douglm@example.org
PERCENT-COMPLETE: 39
END: VTODO
END: VCALENDAR

maps to

{ 
 "@type": "jstask",
 "prodid": "-//ABC Corporation//NONSGML My Product//EN",
 ...
 "participants": {
 "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
  "@type": "Participant",
  "sendTo": {
   "imip": "mailto:douglm@example.org"
  },
  "roles": {
   "attendee": true
  }
 },
 "recurrenceOverrides": {
  "2020-05-23T12:00:00": {
   "participants/be4...6ce3/percentComplete": 39
  },
  ...
 }
 }

4.25. PRIORITY

Simply copy value into the JSCalendar "priority" property.
4.26. PRODID

For a vcalendar JSGroup object with multiple JSEvent and/or JSTask object the [RFC5545] VCALENDAR PRODID is mapped to a JSCalendar "prodid" property in the group.

When mapping to a single JSEvent and/or JSTask object the [RFC5545] VCALENDAR PRODID is mapped to a JSCalendar "prodid" property in the group

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

maps to

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

4.27. RECURRENCE-ID

Refer to Section 5 for information on mapping recurrence ids.

4.28. RELATED-TO

This is mapped to the JSCalendar "relatedTo" property which is a map of relations with the target UID as the keys. The iCalendar relation is by default a PARENT relationship. There is no default for JSCalendar so the relationship must be explicitly specified.

The RELTYPE parameter values map to their lowercase equivalents in the "relation" property.

Also note that the iCalendar relationship types are not identical. CHILD and PARENT map to JSCalendar "child" and "parent" but the best match for iCalendar SIBLING is "next"
RELATED-TO: jsmith.part7.19960817T083000.xyzMail@example.com
RELATED-TO; RELTYPE=SIBLING:
  19960401-080045-4000F192713-0052@example.com
...

maps to

"relatedTo" : {
  "jsmith.part7.19960817T083000.xyzMail@example.com" : {
    "@type" : "Relation",
    "relation" : {
      "parent" : true
    }
  },
  "19960401-080045-4000F192713-0052@example.com" : {
    "@type" : "Relation",
    "relation" : {
      "next" : true
    }
  },
  {
    "@type": "jsevent",
    "prodid": "-//ABC Corporation//NONSGML My Product//EN",
    ...
  }
}

4.29. REQUEST-STATUS

Copy the value into the JSCalendar "requestStatus" property.

4.30. RESOURCES

The RESOURCES property value is a comma-separated list of resources. First split this into the separate resource names and then each resource is mapped on a participant object with a "kind" property of "resource" and the "name" property set to the resource name.

Mapping parameters:

ALTREP  Use the same process as for the ATTENDEE DIR parameter:
create a link property with the "rel" property set to "alternate"
and the "href" property set to the value of the ALTREP parameter.
Then add the link to the participants "links" property.

LANG  Set the participants "language" property.
For an example see Section 6.3

4.31. RDATE

If the RDATE has a RANGE=THISANDFUTURE parameter then the recurrence MUST be split at this RDATE.

Truncate the original object before this RDATE, create a new master representing the object and link them by setting the jscalendar "relatedTo" property in both.

Otherwise create a patch object with the recurrence id set from the RDATE value. If the instance has overrides the differences will also be set in the object.

4.32. RRULE

Each RRULE is converted to an object in the JSCalendar "recurrenceRules" property. Each entry has the type "RecurrenceRule".

...  
RRULE:...
...
maps to...
...
"recurrenceRules" : [{
  "@type" : "RecurrenceRule",
...
}],
...

The recurrence rule object has one property for each element of the recurrence rule. The iCalendar rule has to be parsed out and the individual jscalendar property values set. Most take the same type but there are exceptions.

FREQ (mandatory) Copy into the jscalendar "frequency" property converted to lowercase.

INTERVAL If present and not 1 copy into the jscalendar "interval" property.

RScale If present copy into the jscalendar "rscale" property converted to lowercase.
SKIP  If present copy into the jscalendar "skip" property converted to lowercase.

WKST  If present copy into the jscalendar "firstDayOfWeek" property converted to lowercase.

BYDAY  If present each element becomes an entry in the jsCalendar "byDay" property. This is an array of NDay objects which may have 2 properties:

day  The two character weekday abbreviation.

nthOfPeriod  If the weekday abbreviation is preceded by a signed integer value set the jscalendar "nthOfPeriod" property.

... RRULE:...,BYDAY=-1MO ...

maps to

..."recurrenceRules" : [{
   "@type" : "RecurrenceRule",
   ...
   "byday": [{
      "day": "mo",
      "nthOfPeriod": -1
   }]
   ...
}],
...

Bymonthday  If present each element will be an element in the jscalendar "byMonthDay" property.

Bymonth  If present each element will be an element in the jscalendar "byMonth" property.

Note that the iCalendar values are numeric but the JSCalendar values are strings. This is because of the possible "L" suffix for leap months.

Byyearday  If present each element will be an element in the jscalendar "byYearDay" property.

Byweekno  If present each element will be an element in the jscalendar "byWeekNo" property.
BYHOUR  If present each element will be an element in the jscalendar "byHour" property.

BYMINUTE  If present each element will be an element in the jscalendar "byMinute" property.

BYSECOND  If present each element will be an element in the jscalendar "bySecond" property.

BYSETPOS  If present each element will be an element in the jscalendar "bySetPosition" property.

COUNT  If present set in the jscalendar "count" property.

UNTIL  If present set the jscalendar "until" property with the appropriately reformatted value. If the is no time part append a 0 time and reformat as a jscalendar local date/time.

Some examples:

...  
RRULE:FREQ=DAILY;COUNT=10  
...  

maps to

...  
"recurrenceRules" : [{  
   "@type" : "RecurrenceRule",  
   "frequency" : "daily",  
   "count" : 10  
}]

...
...  
RRULE:FREQ=YEARLY;UNTIL=20220512T140000Z;  
 BYMONTH=1;BYDAY=SU,MO,TU,WE,TH,FR,SA  
...

maps to

...  
"recurrenceRules" : [{  
 "@type" : "RecurrenceRule",  
 "frequency": "yearly",  
 "byMonth": ["1"],  
 "byDay": [{  
   "day": "su"  
  },  
   "day": "mo"  
  },  
   "day": "tu"  
  },  
   "day": "we"  
  },  
   "day": "th"  
  },  
   "day": "fr"  
  },  
   "day": "sa"  
  }],  
 "until": "2022-05-12T10:00:00"  
}],  
...
...<br>RRULE:FREQ=MONTHLY;COUNT=6;BYDAY=-2MO<br>...

maps to

...
"recurrenceRules" : [{
  "@type" : "RecurrenceRule",
  "frequency": "monthly",
  "byDay": [{
    "day": "mo",
    "nthOfPeriod": -2
  }],
  "count": 6
}],
...<br>

4.33. SEQUENCE

Copy the value into the JSCalendar "sequence" property.

4.34. STATUS

For a VEVENT copy the lower-cased value into the JSCalendar "status" property.

For a VTODO copy the lower-cased value into the JSCalendar "progress" property.

4.35. STRUCTURED-DATA

This property is mapped on to a JSCalendar "link" object with the value mapped on to the JSCalendar "href" property in a manner depending on the "STRUCTURED-DATA" "VALUE" parameter:

VALUE=TEXT  Copy the value as a [RFC2397] data uri either as plain text or by encoding as a base64 value. If plain text the value may need escaping as per [RFC2397].

VALUE=BINARY  Copy the value as a [RFC2397] data uri specifying base64 encoding.

VALUE=URI  Copy the value as-is into the href.

The "STRUCTURED-DATA" "SCHEMA" parameter is mapped on to a JSCalendar "schema" property within the link object.
The "STRUCTURED-DATA" "FMTTYPE" parameter is mapped on to a JSCalendar "contentType" property within the link object.

For example:

...

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

...

maps to (with data truncated)

...

"links": {
  "1": {
    "@type": "Link",
    "contentType": "application/ld+json",
    "schema": "https://schema.org/SportsEvent",
    "href": "data:base64;ewogICAgICAgICJAY29udGV4dCI6IIC...
  }
}

...

4.36. SUMMARY

Copy the value into the JSCalendar "title" property.

Mapping parameters:

ALTREP No mapping.

LANG Use the "locale" property.

4.37. TRANSP

If the value of the TRANSP property (ignoring case) is "opaque" set the JSCalendar "freeBusyStatus" property to the value "busy".

Otherwise set the JSCalendar "freeBusyStatus" property to the value "free".
4.38. UID

Copy the value into the JSCalendar "uid" property.

4.39. URL

Maps to a Link object in the JSCalendar object’s "links" property, with the URL property value mapped to the Link "href" property.

5. Translating iCalendar Recurrences

5.1. Translating iCalendar Recurrences: Simple objects with overrides

A simple object with overrides will be converted to a jsCalendar master event with the rules, recurrence dates and exclusion dates translated appropriately.

Overrides MUST be mapped on to a jsCalendar patch object and added to the "recurrenceOverrides" property of the master event with the key being the value of the iCalendar RECURRENCE-ID translated to a json format.

Any override property with the same value as the master SHOULD be omitted. Remaining properties MAY be added in full. Where appropriate, differences SHOULD be expressed as a patch.

This can result in a significant reduction in size for objects with small changes to overrides, for example changing the participation status of an attendee.

5.2. Translating iCalendar Recurrences: Overrides with no master

When inviting an attendee to a single instance of a recurring event, only that override should be sent to the attendee. In this case the override should be a complete jsCalendar object with the type set to the type of the master.

Additionally, there MUST be a recurrenceId property set to the value of the recurrence id for that override. If the timezone of the start of the instance is different from the master value, then there must also be a "recurrenceIdTimeZone" property set to the start timezone of the master.

6. Translating iCalendar: Further examples

This section provides more complete examples of translating from [RFC5545] to JSCalendar.
As usual note that json string values may be split because of line width limits. This is not legal json.

6.1. Recurring event with ATTACH

This is an example of a recurring event with overrides. The first override removes an ATTACH property and adds an ATTACH property. The second override removes all ATTACH properties.

BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//example.org//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200522T120000
DURATION:PT1H
RRULE:FREQ=DAILY;COUNT=8
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBDErecur-1
ATTACH;FMTTYPE=text/plain:http://example.org/doc1.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc2.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc3.txt
END:VEVENT
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200523T120000
DURATION:PT1H
RECURRENCE-ID;TZID=America/New_York:20200523T120000
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBDErecur-1
ATTACH;FMTTYPE=text/plain:http://example.org/doc2.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc3.txt
ATTACH;FMTTYPE=text/plain:http://example.org/doc4.txt
END:VEVENT
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200524T120000
DURATION:PT1H
RECURRENCE-ID;TZID=America/New_York:20200524T120000
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBDErecur-1
END:VEVENT
END:VCALENDAR

maps to

```json
{

}
```
"prodId": "/example.org//EN",
"entries": [
  {
    "links": {
      "1": {
        "@type": "Link",
        "rel": "enclosure",
        "contentType": "text/plain",
        "href": "http://example.org/doc1.txt"
      },
      "2": {
        "@type": "Link",
        "rel": "enclosure",
        "contentType": "text/plain",
        "href": "http://example.org/doc2.txt"
      },
      "3": {
        "@type": "Link",
        "rel": "enclosure",
        "contentType": "text/plain",
        "href": "http://example.org/doc3.txt"
      }
    },
    "created": "2020-05-23T17:04:50Z",
    "start": "2020-05-22T12:00:00",
    "timeZone": "America/New_York",
    "duration": "PT1H",
    "title": "recurring daily 8 times",
    "uid": "6252D6C40A8308BFE25BBDErecur-1",
    "recurrenceRules": [
      {
        "@type": "RecurrenceRule",
        "frequency": "daily",
        "count": 8
      }
    ],
    "recurrenceOverrides": {
      "2020-05-23T12:00:00": {
        "recurrenceId": "2020-05-23T12:00:00",
        "links/d4a618d4-929c-4c81-ae5b-322afe407a00": null,
        "links/fb75b76a-a159-4a86-bd3d-7ace6b39c6c3": {
          "@type": "Link",
          "rel": "enclosure",
          "contentType": "text/plain",
          "href": "http://example.org/doc4.txt"
        }
      },
      "2020-05-24T12:00:00": {

6.2. Simple event with CONTACT

This example shows the conversion of a simple event with a single CONTACT property in JSCalendar.

BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//Example//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:20200522T142047Z
DTSTART;TZID=America/New_York:20200622T120000
DURATION:PT1H
SUMMARY:event with contact
UID:6252D6C40A8308BFE25BBEFcontact-1
CONTACT;ALTREP="ldap://example.com:6666/o=ABC%20Industries,c=US???(cn=Jim%20Dolittle)";Jim Dolittle, ABC Industries, +1-919-555-1234
END:VEVENT
END:VCALENDAR

translates to

{  
  "@type": "jsgroup",
  "prodId": "//Example.org//Example V3.13.2//EN",
  "entries": [
    {  
      "@type": "jsevent",
      "participants": {
        "40288108-733187c1-0173-3188007b-00000001": {
          "@type": "Participant",
          "roles": {
            "contact": true
          },
          "description": "Jim Dolittle, ABC Industries, +1-919-555-1234",
          "links": ["recurrenceId": "2020-05-24T12:00:00",
            "links/d4a618d4-929c-4c81-ae5b-322afe407a00": null,
            "links/6c54e72e-3413-487c-ae14-fb318a90db43": null,
            "links/44087e9a-132c-4a5d-b25d-4ce580ed8004": null
          ]
        }
      }
    }
  ]
}
6.3. Simple event with RESOURCES

TBD

6.4. Recurring event. Attendees only in overrides

In this more complex example there is no ORGANIZER or ATTENDEES in the master event. There are overrides which invite one or more attendees.

For one override the ORGANIZER is also an ATTENDEE. In the other that is not the case. This is reflected in the "roles" property for the organizer.

Note that each override has its own "participants" property and the first has a links property to handle the DIR parameter on one attendee.
RRULE:FREQ=DAILY;COUNT=8
END:VEVENT
BEGIN:VEVENT
RECURRENCE-ID;TZID=America/New_York:20200523T120000
ATTENDEE:mailto:douglm@example.org
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2;DIR="http://example.org/vcards/vbede.vcf":mailto:vbede@example.org
CREATED:20200704T035515Z
DURATION:PT1H
DTSTART;TZID=America/New_York:20200523T120000
LAST-MODIFIED:20200704T035706Z
ORGANIZER:mailto:douglm@example.org
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
BEGIN:VEVENT
RECURRENCE-ID;TZID=America/New_York:20200524T120000
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2:mailto:user01@example.org
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2:mailto:vbede@example.org
CREATED:20200704T035515Z
DURATION:PT1H
DTSTART;TZID=America/New_York:20200524T120000
LAST-MODIFIED:20200704T035706Z
ORGANIZER:mailto:douglm@example.org
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
END:VCALENDAR

translates to

{  
  "@type": "jsgroup",
  "prodId": "//Example.org//Example V3.13.2//EN",
  "entries": [
    {  
      "@type": "jsevent",
      "created": "2020-07-04T03:57:06Z",
      "start": "2020-05-23T12:00:00",
      "timeZone": "America/New_York",
      "duration": "PT1H",
      "title": "recurring daily 8 times",
      "uid": "6252D6C40A8308BFE25BBEFrecur1-1",
      "recurrenceRules": [
        {  
          "@type": "RecurrenceRule",
          "freq": "DAILY",
          "count": 8
        }
      ]
    }
  ]
}
"frequency": "daily",
"count": 8
}
],
"recurrenceOverrides": {
  "2020-05-23T12:00:00": {
    "participants": {
      "be450b70-9bf7-4f6e-8f65-971ed566ce3": {
        "@type": "Participant",
        "sendTo": {
          "imip": "mailto:douglm@example.org"
        },
        "roles": {
          "attendee": true,
          "owner": true
        }
      },
      "a539dfe3-4463-4f28-b9de-17d3a0e99faf": {
        "@type": "Participant",
        "sendTo": {
          "imip": "mailto:vbede@example.org"
        },
        "expectReply": true,
        "links": {
          "1": {
            "@type": "Link",
            "href": "http://example.org/vcards/vbede.vcf",
            "rel": "alternate"
          }
        },
        "roles": {
          "attendee": true
        },
        "scheduleStatus": "1.2"
      }
    },
    "replyTo": {
      "imip": "mailto:douglm@example.org"
    }
  },
  "2020-05-24T12:00:00": {
    "participants": {
      "daee4cf-6f6a-4ce3-9f4d-6bd884650d3d": {
        "@type": "Participant",
        "sendTo": {
          "imip": "mailto:user01@example.org"
        },
        "expectReply": true,
7. Translating JSCalendar properties to iCalendar components

This section lists JSCalendar property types that map on to [RFC5545] components.

7.1. jsevent

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

If it is a single VEVENT then a [RFC5545] VCALENDAR component must surround it and the JSCalendar "prodid" property will be converted to a [RFC5545] PRODID.
When converting multiple JSEvent or JSTask objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the JSGroup "prodid" or generated.

7.2. jsgroup

A JSCalendar object with a type of "jsgroup" is mapped on to a [RFC5545] VCALENDAR component.
{  
  "@type": "jsgroup",
  "prodid": "-//ABC Corporation//NONSGML My Product//EN",
  ...
  
  {  
    "@type": "jsevent",
    ...
  }
  
  {  
    "@type": "jsevent",
    ...
  }
}

maps to

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

7.3. jstask

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

If it is a single VTODO then a [RFC5545] VCALENDAR component must surround it and the JSCalendar "prodid" property will be converted to a [RFC5545] PRODID.
maps to

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

When converting multiple JSEvent or JSTask objects the surrounding
[RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from
either the JSGroup "prodid" or generated.

7.4. timezones

The JSCalendar TimeZone objects within a "timezones" property are
mapped on to [RFC5545] VTIMEZONE components within the surrounding
VCALENDAR component. Each mapped TimeZone MUST only appear once.
{  "@type": "jsevent",  ...  "timezones": {  "/Example/Somewhere": {  "@type": "TimeZone",  ...  },  "/Example/Somewhere-else": {  "@type": "TimeZone",  ...  }  }  }

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

When converting multiple JSEvent or JSTask objects the surrounding
[RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from
either the JSGroup "prodid" or generated.

7.5. alerts

Each member of a JSCalendar alerts property maps to a [RFC5545]
VALARM component. Only display and email alarms are allowed in
JSCalendar.

7.5.1. action

The JSCalendar "alert" property maps to the [RFC5545] ACTION
property. The value SHOULD be the uppercased version if the
JSCalendar "alert" property.

For example:
...  
"action": "display",  
...  
maps to  
ACTION:DISPLAY  
and  
...  
"action": "email",  
...  
maps to  
ACTION:EMAIL  

7.5.2. trigger  

A JSCalendar trigger with a type of "AbsoluteTrigger" maps on to a [RFC5545] TRIGGER property with a "VALUE" parameter of "DATE-TIME" and a value taken from the JSCalendar "when" property.  

For example:  

"trigger": {  
   "@type": "AbsoluteTrigger",  
   "when": "20210315T133000Z"  
  }  

maps to  
TRIGGER;VALUE=DATE-TIME:20210315T133000Z  

A JSCalendar trigger with a type of "OffsetTrigger" maps on to a [RFC5545] TRIGGER property with a duration value taken from the JSCalendar "offset" property.  

If the JSCalendar trigger has a "relativeTo" property with the value "end" then the [RFC5545] TRIGGER property will have a RELATED=END parameter.  

For example:
"trigger": {
    "@type": "OffsetTrigger",
    "offset": "-P2D",
    "relativeTo": "end"
}

maps to
TRIGGER;RELATED=END:-P2D

and

"trigger": {
    "@type": "OffsetTrigger",
    "offset": "-PT30M"
}

maps to
TRIGGER:-PT30M

7.5.3. todo

Need to deal with "acknowledged" and "relatedTo". Also in the icalendar to jscalendar.

8. Translating JSCalendar properties to iCalendar properties

This section is an alphabetic list of all JSCalendar property types that map on to [RFC5545] properties.

8.1. categories

Each member of the JSCalendar "categories" property maps on to a [draft-ietf-calext-eventpub-extensions] CONCEPT property with the value being the key of each member.

For example:
"categories": {
  "http://example.com/event-types/arts/music": true,
  "http://example.com/performance-types/arts/live": true
}

maps to

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

8.2. created

The JSCalendar "created" property maps on to a [RFC5545] CREATED property with the value being the [RFC5545] UTC date-time derived from the value of the property.

For example:

...  "created": "2021-03-15T13:30"00Z"
...

maps to

...  CREATED:20210315T133000Z
...

8.3. duration

The JSCalendar "duration" property is only valid for event objects. Copy the JSCalendar "duration" property in to the [RFC5545] DURATION property.

For example:
...  
"duration": "PT1H"
...  
maps to

...  
DURATION:PT1H
...  

8.4. estimatedDuration

The JSCalendar "estimatedDuration" property is only valid for task objects. Copy the JSCalendar "estimatedDuration" property in to the [RFC5545] ESTIMATED-DURATION property.

For example:

...  
"estimatedDuration": "PT18H"
...  
maps to

...  
ESTIMATED-DURATION:PT18H
...  

8.5. keywords

Each member of the JSCalendar "categories" property maps on to a [RFC5545] CATEGORIES property with the value being the key of each member.

For example:
... 
"keywords": {
  "APPOINTMENT": true,
  "EDUCATION": true,
  "MEETING": true
},
...

maps to

...
CATEGORIES:APPOINTMENT
CATEGORIES:EDUCATION
CATEGORIES:MEETING
...

or alternatively

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

8.6. locations

JSCalendar locations should be mapped on to [draft-ietf-calext-eventpub-extensions]VLOCATION components. Additionally, for backwards compatibility, a location should be mapped on to a [RFC5545] LOCATION property. This property should be mapped from the only location or the one related to the start.

8.7. participants

JSCalendar participants will be mapped on to different iCalendar properties and components depending on their jsCalendar role values.

A participant with a role containing "contact" MUST be mapped on to an iCalendar CONTACT property and SHOULD also be mapped on to a [draft-ietf-calext-eventpub-extensions]PARTICIPANT component which provides a better mapping.

A participant with a role containing "owner" MUST be mapped on to an iCalendar ORGANIZER property and SHOULD also be mapped on to a [draft-ietf-calext-eventpub-extensions]PARTICIPANT component which provides a better mapping.

A participant with a role containing any of "attendee", "optional" or "informational" MUST be mapped on to an iCalendar ATTENDEE property and SHOULD also be mapped on to a
[draft-ietf-calext-eventpub-extensions]PARTICIPANT component which provides a better mapping.

A more complete mapping may be achieved by creating a [draft-ietf-calext-eventpub-extensions]PARTICIPANT component.

For all properties the participants jsCalendar "language" property, if present, is mapped on to the iCalendar "LANG" property parameter.

For all properties if the participant contains a jsCalendar "link" with a "rel" of "alternate" then the value of the link is used for the iCalendar "ALTREP" property parameter.

Where do we get the cua?

9. Security Considerations

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

10. IANA Considerations

None.

11. 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.

12. References

12.1. Normative References

[draft-ietf-calext-eventpub-extensions]
"Event Publishing Extensions to iCalendar",

[draft-ietf-calext-ical-relations]
"Support for iCalendar Relationships",
12.2. Informative References

[draft-apthorp-ical-tasks]
"Task Extensions to iCalendar",

[draft-ietf-calext-valarm-extensions]
"VALARM Extensions for iCalendar",


Appendix A.  Outdated document sections

A.1.  Note

The sections following this one are all the original ones from draft 1 written by Robert/Neil - there for reference.

A.2.  JSTask

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

<table>
<thead>
<tr>
<th>Property</th>
<th>iCalendar counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>due</td>
<td>Maps to the DUE property. See Appendix A.4.1.</td>
</tr>
<tr>
<td>statusUpdatedAt</td>
<td>COMPLETED property. The JSTask status property MUST have value &quot;completed&quot;.</td>
</tr>
<tr>
<td>progress</td>
<td>PARTSTAT and COMPLETED properties, including the definitions in the RFC draft [draft-apthorp-ical-tasks].</td>
</tr>
<tr>
<td>status</td>
<td>STATUS property, including the definitions in the RFC draft [draft-apthorp-ical-tasks].</td>
</tr>
</tbody>
</table>

Table 1: Mapping JSTask properties

A.3.  JSGroup

A JSGroup maps to a iCalendar VCALENDAR containing VEVENT or VTODO components.
### Table 2: Mapping JSGroup properties

#### A.4. Common properties

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

<table>
<thead>
<tr>
<th>Property</th>
<th>iCalendar counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>@type</td>
<td>Determined by the iCalendar component type: &quot;jsevent&quot; for VEVENT, &quot;jstask&quot; for VTODO, &quot;jsgroup&quot; for VCALENDAR.</td>
</tr>
<tr>
<td>alerts</td>
<td>Each entry maps to a VALARM component. The action property maps to iCalendar ACTION, where both iCalendar &quot;DISPLAY&quot; and &quot;AUDIO&quot; values map to the &quot;display&quot; action. An EMAIL value maps to a JSCalendar &quot;email&quot; action. <em>relativeTo</em> and <em>offset</em> map to the TRIGGER property.</td>
</tr>
<tr>
<td>color</td>
<td>COLOR property, as specified in [RFC7986].</td>
</tr>
<tr>
<td>created</td>
<td>CREATED property.</td>
</tr>
<tr>
<td>description</td>
<td>DESCRIPTION property.</td>
</tr>
<tr>
<td>descriptionContentType</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>excluded</td>
<td>EXDATE property.</td>
</tr>
<tr>
<td>freeBusyStatus</td>
<td>TRANSP property.</td>
</tr>
<tr>
<td>invitedBy</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>keywords</td>
<td>CATEGORIES property, as specified in [RFC7986].</td>
</tr>
<tr>
<td>links</td>
<td>ATTACH ([RFC5545]), URL or IMAGE ([RFC7986]) properties with URI value types map to the the Link <em>href</em>. The FMTTYPE parameter maps to <em>type</em>, the SIZE parameter to <em>size</em>. Mapping other properties is implementation-specific.</td>
</tr>
<tr>
<td>locale</td>
<td>LANGUAGE parameter of the SUMMARY or DESCRIPTION property.</td>
</tr>
<tr>
<td>localizations</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>locations</td>
<td>See Appendix A.4.2.</td>
</tr>
<tr>
<td>method</td>
<td>METHOD property of the embedding VCALENDAR.</td>
</tr>
<tr>
<td>participants</td>
<td>See Appendix A.4.3.</td>
</tr>
<tr>
<td>priority</td>
<td>PRIORITY property.</td>
</tr>
<tr>
<td>privacy</td>
<td>CLASS property.</td>
</tr>
<tr>
<td>prodId</td>
<td>PRODID property.</td>
</tr>
<tr>
<td>recurrenceOverrides</td>
<td>RDATE and EXDATE properties, and any VEVENT or VTODO instances with a recurrence-id and same UID as the mapped main object.</td>
</tr>
<tr>
<td>recurrenceRule</td>
<td>RRULE property. For all-day calendar objects, map the <em>until</em> property value to an iCalendar DATE (effectively removing the time component). To convert a DATE-typed UNTIL from iCalendar, set the time components of the LocalDateTime value to &quot;23:59:59&quot;. If the iCalendar UNTIL value is a UTC date time, convert it to the local time in the JSCalendar calendar object time zone. To convert to iCalendar where the DTSTART or DUE property is of type DATE, omit the time component of the LocalDateTime value.</td>
</tr>
<tr>
<td>relatedTo</td>
<td>RELATED-TO property.</td>
</tr>
<tr>
<td>Property</td>
<td>iCalendar Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>replyTo</td>
<td>An iCalendar ORGANIZER with a mailto: URI mapped to the &quot;imip&quot; method, or any other URI mapped to the &quot;other&quot; method. Mapping multiple methods is implementation-specific.</td>
</tr>
<tr>
<td>sequence</td>
<td>SEQUENCE property.</td>
</tr>
<tr>
<td>showWithoutTime</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>start</td>
<td>Maps to the DTSTART property. See Appendix A.4.1.</td>
</tr>
<tr>
<td>status</td>
<td>STATUS property.</td>
</tr>
<tr>
<td>timeZone</td>
<td>Maps to the TZID parameter. See Appendix A.4.1.</td>
</tr>
<tr>
<td>timeZones</td>
<td>Each entry in the property maps to a VTIMEZONE in the embedding VCALENDAR component.</td>
</tr>
<tr>
<td>title</td>
<td>SUMMARY property.</td>
</tr>
<tr>
<td>uid</td>
<td>UID property.</td>
</tr>
<tr>
<td>updated</td>
<td>DTSTAMP and LAST-MODIFIED properties.</td>
</tr>
<tr>
<td>useDefaultAlerts</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>virtualLocations</td>
<td>See Appendix A.4.2.</td>
</tr>
</tbody>
</table>

Table 3: Translation between JSCalendar and iCalendar

A.4.1. Time properties and types

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

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

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

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

o The "showWithoutTime" property value is not set or "true".

o The recurrenceRule.frequency is at most daily.

o All recurrenceOverrrides recurrence ids have zero time

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

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

o With UTC time, if the "timeZone" property value equals "Etc/UTC".

o With local time, if the "timeZone" property value is null or not set.

A.4.2. Locations

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

<table>
<thead>
<tr>
<th>Property</th>
<th>iCalendar counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordinates</td>
<td>GEO property.</td>
</tr>
<tr>
<td>description</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>name</td>
<td>LOCATION property value.</td>
</tr>
<tr>
<td>rel</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>timeZone</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>uri</td>
<td>The LOCATION ALTREP parameter.</td>
</tr>
</tbody>
</table>

Table 4: Mapping Location properties

The iCalendar counterpart for JSCalendar VirtualLocation objects is the iCalendar [RFC7986] CONFERENCE property.
<table>
<thead>
<tr>
<th>Property</th>
<th>iCalendar counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>name</td>
<td>LABEL parameter.</td>
</tr>
<tr>
<td>uri</td>
<td>CONFERENCE property value.</td>
</tr>
</tbody>
</table>

Table 5: Mapping virtualLocation properties

A.4.3. Participants

The following table outlines translation of JSCalendar participants. An iCalendar ORGANIZER maps to both the replyTo property and a participant with role "owner". If an ATTENDEE with the same CAL-ADDRESS value exists, then it maps to the same participant as the ORGANIZER participant. Other participants map to ATTENDEEs.
<table>
<thead>
<tr>
<th>Property</th>
<th>iCalendar counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>attendance</td>
<td>ROLE parameter values REQ-PARTICIPANT, OPT-PARTICIPANT and NON-PARTICIPANT.</td>
</tr>
<tr>
<td>delegatedFrom</td>
<td>DELEGATED-FROM parameter</td>
</tr>
<tr>
<td>delegatedTo</td>
<td>DELEGATED-TO parameter</td>
</tr>
<tr>
<td>email</td>
<td>EMAIL parameter, if defined. Otherwise the CAL-ADDRESS property value, if it is a mailto: URI.</td>
</tr>
<tr>
<td>expectReply</td>
<td>RSVP parameter</td>
</tr>
<tr>
<td>kind</td>
<td>CUTYPE parameter</td>
</tr>
<tr>
<td>locationId</td>
<td>Implementation-specific.</td>
</tr>
<tr>
<td>memberOf</td>
<td>MEMBER parameter</td>
</tr>
<tr>
<td>name</td>
<td>CN parameter</td>
</tr>
<tr>
<td>participationStatus</td>
<td>PARTSTAT parameter</td>
</tr>
<tr>
<td>roles</td>
<td>ROLE parameter.</td>
</tr>
<tr>
<td>scheduleSequence</td>
<td>SEQUENCE property of the participant’s latest iMIP message</td>
</tr>
<tr>
<td>scheduleUpdated</td>
<td>DTSTAMP property of the participant’s latest iMIP message</td>
</tr>
<tr>
<td>sendTo</td>
<td>A CAL-ADDRESS with a mailto: URI maps to the JSCalendar &quot;imip&quot; method, any other URI to the &quot;other&quot; method. Mapping multiple methods is implementation-specific.</td>
</tr>
</tbody>
</table>

Table 6: Mapping Participant properties

A.5. Custom properties

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

Authors’ Addresses

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

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

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

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

Michael Douglass
Bedework Commercial Services
226 3rd Street
Troy, NY 12180
United States of America

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 27 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
# Table of Contents

1. Acknowledgements ........................................ 2
2. Introduction ............................................ 3
   2.1. Terms and Definitions ............................... 3
3. Discovering alternative access methods .................. 3
4. Enhanced GET ............................................. 4
   4.1. General ........................................... 4
   4.2. Deletions .......................................... 5
   4.3. Handling of invalid sync tokens .................... 5
   4.4. Paging the response ................................ 5
   4.5. Caching of responses ............................... 6
   4.6. Examples ........................................... 6
5. Changes to the iCalendar specifications ................. 8
   5.1. Redefined Status property .......................... 8
6. Header Field: Sync-Token ................................. 10
7. New Prefer header field preferences ..................... 10
   7.1. Preference subscribe-enhanced-get .................. 10
   7.2. Preference limit ................................... 10
8. Link relations ............................................ 10
   8.1. General ........................................... 10
   8.2. subscribe-caldav .................................. 11
   8.3. subscribe-caldav-auth .............................. 11
   8.4. subscribe-webdav-sync .............................. 11
   8.5. subscribe-enhanced-get ............................ 11
9. Security Considerations .................................. 11
10. IANA Considerations ...................................... 12
    10.1. Sync-Token HTTP Header Field Registration ........ 12
    10.2. Preference Registrations ........................... 12
    10.3. Link Relation Registrations ....................... 12
11. Normative References .................................... 13
    Appendix A. Open issues ................................ 14
    Appendix B. Change log ................................ 14
    Author’s Address ...................................... 14

# 1. Acknowledgements

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

Marten Gajda, Ken Murchison, Garry Shutler

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

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

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

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

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

Given that many clients and servers already support CalDAV this provides an easy upgrade path for those clients. Additionally an enhanced GET protocol is specified here to allow a light weight implementation.

The use of subscription upgrade may help reduce load on servers, but perhaps more importantly it allows mobile devices to use a more efficient update mechanism reducing data transferred and presumably improving battery life.

2.1. Terms and Definitions

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

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

3. Discovering alternative access methods

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

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

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

>> Request <<

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

>> Response <<

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

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

4. Enhanced GET

4.1. General

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

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

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

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

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

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

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

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

On subsequent fetches the entity will not be returned.

4.3.  Handling of invalid sync tokens

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

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

4.4.  Paging the response

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

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

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

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

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

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

* subscribe-enhanced-get

* limit

4.6. Examples

EXAMPLE 1

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

>> Request <<

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

>> Response <<

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

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

EXAMPLE 2

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

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

>> Response <<

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

EXAMPLE 3

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

>> Request <<

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

>> Response <<

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

EXAMPLE 4

This is an example of the subsequent request and response when changes have occurred.
5. Changes to the iCalendar specifications

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

5.1. Redefined Status property

Property name STATUS

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

Value Type TEXT

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

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

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

Format Definition

This property is defined by the following notation:

```
status          = "STATUS" statparam ":" statvalue CRLF
statparam       = *(";" other-param)
statvalue       = (statvalue-event
                   / statvalue-todo
                   / statvalue-jour)
statvalue-event = "TENTATIVE" ;Indicates event is tentative.
                   / "CONFIRMED" ;Indicates event is definite.
                   / "CANCELLED" ;Indicates event was cancelled.
                   / "DELETED"  ;Indicates event was deleted.
                   ;Status values for a "VEVENT"
statvalue-todo  = "NEEDS-ACTION" ;Indicates to-do needs action.
                   / "COMPLETED" ;Indicates to-do completed.
                   / "IN-PROCESS" ;Indicates to-do in process of.
                   / "CANCELLED" ;Indicates to-do was cancelled.
                   / "DELETED"  ;Indicates to-do was deleted.
                   ;Status values for "VTODO".
statvalue-jour  = "DRAFT"     ;Indicates journal is draft.
                   / "FINAL"     ;Indicates journal is final.
                   / "CANCELLED" ;Indicates journal is removed.
                   / "DELETED"   ;Indicates journal was deleted.
                   ;Status values for "VJOURNAL".
```

Example

**EXAMPLE 1**

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

```
STATUS:TENTATIVE
```

**EXAMPLE 2**

The following is an example of this property for a "VTODO" calendar component:
EXAMPLE 3

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

6. Header Field: Sync-Token

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

Accept = DQUOTE URI DQUOTE

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

EXAMPLE

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

Sync-Token: "data:,1234567"

7. New Prefer header field preferences

7.1. Preference subscribe-enhanced-get

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

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

7.2. Preference limit

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

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

8. Link relations

8.1. General

This clause defines a number of new link relations required to facilitate subscription upgrades.
8.2. subscribe-caldav

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

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

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

8.3. subscribe-caldav-auth

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

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

8.4. subscribe-webdav-sync

This specifies an access point which supports only webdav sync.

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

The client MUST follow that specification.

8.5. subscribe-enhanced-get

This specifies an access point which supports something new.

The client MUST follow that specification.

9. Security Considerations

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

Privacy Considerations
Properties with a "URI" value type can expose their users to privacy leaks as any network access of the URI data can be tracked. Clients SHOULD NOT automatically download data referenced by the URI without explicit instruction from users. This specification does not introduce any additional privacy concerns beyond those described in [RFC5545].

10. IANA Considerations

10.1. Sync-Token HTTP Header Field Registration

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

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

Figure 1

10.2. Preference Registrations

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

Preference subscribe-enhanced-get
Value None.
Description Marks the interaction as enhanced get and provides the optional sync-token and page size.
Reference this document

Preference limit
Value An integer page size.
Description Provide a limit on the number of components in the response.
Reference this document

10.3. Link Relation Registrations

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

Douglass Expires 27 January 2022 [Page 12]
<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscribe-caldav</td>
<td>Current</td>
<td>Section 8.2</td>
</tr>
<tr>
<td>subscribe-caldav_auth</td>
<td>Current</td>
<td>Section 8.3</td>
</tr>
<tr>
<td>subscribe-webdav-sync</td>
<td>Current</td>
<td>Section 8.4</td>
</tr>
<tr>
<td>subscribe-enhanced_get</td>
<td>Current</td>
<td>Section 8.5</td>
</tr>
</tbody>
</table>

Table 1

11. Normative References


Appendix A. Open issues

Vary  Ensure we get that right.

Appendix B. Change log

calext00 2019-06-05 MD
*  First calext version
*  Use Sync-Token header rather than parameter
v04 2019-03-07 MD
*  Reference to RFC 6538 – WebDAV sync and RFC 7240 Prefer
*  Go back to HEAD
*  New Preference and parameters.
*  Examples
*  More text for extended get. Talk about deletions.
v01 2017-02-17 MD
*  Add text about OPTIONS
*  Add text about read/write CalDAV
v00 2017-02-15 MD
*  First pass

Author’s Address

Michael Douglass

Email: mdouglass@bedework.com