Support for iCalendar Relationships  
draft-ietf-calext-ical-relations-11

Abstract

This specification updates the iCalendar RELATED-TO property defined in RFC5545 by adding new relation types 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 23 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1. Introduction

iCalendar entities defined in [RFC5545] 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 iCalendar [RFC5545] RELATED-TO property has no support for temporal relationships as used by 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 all related to the referring, aggregating component and the retrieval of components based on this key. For example, this may be used to identify the tasks associated with a given project without having to communicate the task structure of the project. A further example is the grouping of all sub-tasks associated with the delivery of 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. These values have some meaning to those who apply them but not necessarily to any consumer. 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 [RFC8288] which defines the generic concept of Web Linking as well as its expression in the HTTP LINK header field.

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.

Calendar components are often grouped into collections to represent a calendar or a series of tasks, for example [RFC4791]’ (CalDAV) calendar collections.

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] has been extended by [RFC8607] to handle server-side management and stripping of inline data and to provide additional data about the attachment (size, filename etc).

Additionally clients may choose to handle attachments differently from the LINK property as attachments are often an integral part of the message - for example, the agenda.

1.5. Caching and offline use

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

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

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.

The notation used in this memo to (re-)define iCalendar elements is the ABNF notation of [RFC5234] as used by [RFC5545]. Any syntax elements shown below that are not explicitly defined in this specification come from iCalendar [RFC5545].

2. LINK Property Reference Types

The reference value in the LINK property defined below can take three forms specified by the VALUE parameter:

URI: This is a URI referring to the target.
UID: This allows for linking within a single collection of calendar components and the value MUST refer to another component within the same collection.

XML-REFERENCE: In an XML environment it may be necessary to refer to a fragment of an external XML artifact. This value is a URI with an XPointer anchor value. The XPointer is defined in [W3C.WD-xptr-xpointer-20021219] and its use as an anchor is defined in [W3C.REC-xptr-framework-20030325]

Note that UID references may need updating on import. An example, is data to be imported from a file containing VTODO and VEVENT components with a VTODO referring to VEVENT components by UID. When imported into a CalDAV system, the VTODO components are typically placed in a different collection from the VEVENT components. This would require the UID reference to be replaced with a URI.

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 (a negative value) or lag (a positive value) 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. This is indicated by the direction of the arrow in the diagrams below.
Also each relationship may be modified by the addition of a GAP parameter to the relationship which applies to the targeted component.

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

```
+----------+
| Task-A   |
+----------+
    V
+----------+
| Task-B   |
+----------+
```

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, in the development of two related pieces of software, e.g. the api and the implementation, the design of the implementation (B) cannot be completed until the design of the api (A) has been completed.

```
+----------+
| Task-A   |-->+
+----------+
    |
+----------+
| Task-B   |<--+
+----------+
```

Figure 2: Finish to finish relationship

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

```
+----------+
| Task-A   |
+----------+
    |
+----------+
| Task-B   |<--+
+----------+
```
5. Additional New RELTYPE Parameter Values

This section defines the additional relationships below:

RELTYPE=FIRST: Indicates that the referenced calendar component is the first in a series the referencing calendar component is part of.

RELTYPE=NEXT: Indicates that the referenced calendar component is the next in a series the referencing 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.

Note that the relationship types of PARENT, CHILD and SIBLING establish a hierarchical relationship. The new types of FIRST and NEXT are an ordering relationship.

6. New Property Parameters
6.1. Link Relation

Parameter name: LINKREL

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

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

```
linkrelparam = "LINKREL" "="
  ("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. There is no default relation type.

In addition to the value defined here any link relation in the link registry established by [RFC8288], or new link relations, may be used.

It is expected that link relation types seeing significant usage in calendaring will have the calendaring usage described in an RFC.

LINKREL=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 components with a temporal relationship.

Format Definition: This parameter is defined by the following notation where dur-value is defined in section 3.3.6 of [RFC5545].

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

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

An example of lag time might be if task A is "paint the room" and task B is "lay the carpets" then task A may be related to task B with RELTYPE=FINISHTOSTART with a gap of 1 day - long enough for the paint to dry.

```
| Paint the room  |--+
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(lag of one day)</td>
</tr>
<tr>
<td>+--</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Figure 5: Finish to start relationship with lag

For an example of lead time, in constructing a two storey building the electrical work must be done before painting. However the painter can move in to the first floor as the electricians move upstairs.

```
| Electrical work  |--+
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(lead of estimated time)</td>
</tr>
<tr>
<td>+--</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Figure 6: Finish to start relationship with lead

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.

**XML-REFERENCE** VALUE=XML-REFERENCE indicates that the associated
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.

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: https://example.com/event-types/arts/music

8.2. Link

Property name: LINK

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

Value type: URI, UID or XML-REFERENCE

Property Parameters: The VALUE parameter is required. Non-standard, link relation type, format type, label and language 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 ":" 
     ( uri / ; for VALUE=XML-REFERENCE 
       uri / ; for VALUE=URI 
       text ) ; for VALUE=UID 
CRLF

linkparam = ; the elements herein may appear in any order, 
            ; and the order is not significant. 
            (; "VALUE" "=" ("XML-REFERENCE" / 
              "URI" / 
              "UID")) 
           1*(; "linkrelparam") 
           1*(; "fmttypeparam") 
           1*(; "labelfparam") 
           1*(; "languageparam") 
           *(; "other-param")
```
This property is a serialisation of the model in [RFC8288], where the link target is carried in the property value, the link context is the containing calendar entity, and the link relation type and any target attributes are carried in iCalendar property parameters.

The LINK property parameters map to [RFC8288] attributes as follows:

- **LABEL**: Maps to the "title" attribute defined in section 3.4.1 of [RFC8288].
- **LANGUAGE**: Maps to the "hreflang" attribute defined in section 3.4.1 of [RFC8288].
- **LINKREL**: Maps to the link relation type defined in section 2.1 of [RFC8288].
- **FMTTYPE**: Maps to the "type" attribute defined in section 3.4.1 of [RFC8288].

There is no mapping for [RFC8288] "title*", "anchor", "rev" or "media".

**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: https://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: https://example.com/tasks/01234567-abcd1234.ics
```

**Example**: The following is an example of this property which provides a reference to a fragment of an XML document. The link relation is a vendor defined value.

```
LINK;LINKREL="https://example.com/linkrel/costStructure";
VALUE=XML-REFERENCE:
https://example.com/xmlDocs/bidFramework.xml
#xpointer(descendant::CostStruc/range-to(following::CostStrucEnd[1]))
```
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)

The current link registry

Example: The following is an example of this property.

REFID:itinerary-2014-11-17

9. Updates to RFC 5545

This specification updates the RELATED-TO property defined in Section 3.8.4.5 of [RFC5545]. The contents of Section 9.1 replace that section.

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.
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.

The FIRST and NEXT define ordering relationships between calendar components.

The DEPENDS-ON relationship 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.

The REFID and CONCEPT relationships establish a reference from the current component to the referenced component.

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.

Deletion of the target component, for example the target of a FIRST, NEXT or temporal relationship can result in broken links.

It is up to the target calendar system to maintain any property implications of these relationships.

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

```
related    = "RELATED-TO" relparam ":"
            ( text / ; 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.
10. Security Considerations

All of the security considerations of section 7 of [RFC5545] apply to this specification.

Applications using the LINK property need to be aware of the risks entailed in using the URIs provided as values. See section 7 of [RFC3986] for a discussion of the security considerations relating to URIs.

In particular note section 7.1 "Reliability and Consistency" of [RFC3986] which points out the lack of a stability guarantee for referenced resources.

When the value is an XML-REFERENCE type the targeted data is an XML document or portion thereof. Consumers need to be aware of the security issues related to XML processing - in particular those related to XML entities. See [RFC4918] - Section 20.6. Additionally note that the reference may be invalid or become so over time.

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

Extremely large values for the GAP parameter may lead to unexpected behavior.

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>LINKREL</td>
<td>Current</td>
<td>Section 6.1</td>
</tr>
</tbody>
</table>

Table 2

11.3. iCalendar Value Data Type Registrations

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

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

Table 3
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].

<table>
<thead>
<tr>
<th>Relationship Type</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCEPT</td>
<td>Current</td>
<td>Section 5</td>
</tr>
<tr>
<td>DEPENDS-ON</td>
<td>Current</td>
<td>Section 5</td>
</tr>
<tr>
<td>FINISHTOFINISH</td>
<td>Current</td>
<td>Section 4</td>
</tr>
<tr>
<td>FINISHTOSTART</td>
<td>Current</td>
<td>Section 4</td>
</tr>
<tr>
<td>FIRST</td>
<td>Current</td>
<td>Section 5</td>
</tr>
<tr>
<td>NEXT</td>
<td>Current</td>
<td>Section 5</td>
</tr>
<tr>
<td>REFID</td>
<td>Current</td>
<td>Section 5</td>
</tr>
<tr>
<td>STARTTOFINISH</td>
<td>Current</td>
<td>Section 4</td>
</tr>
<tr>
<td>STARTTOSTART</td>
<td>Current</td>
<td>Section 4</td>
</tr>
</tbody>
</table>

Table 4

11.5. New Reference Type Registration

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

<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>

Table 5
12. Acknowledgements

The author would like to thank the members of CalConnect, 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
United States of America
Email: mdouglass@bedework.com
URI:   https://bedework.com

Douglass                Expires 23 September 2022              [Page 21]
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 22 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1. Acknowledgements .................................................. 3
2. Introduction .......................................................... 3
   2.1. Terms and Definitions ........................................ 4
3. Task Architecture .................................................... 5
4. Task Architecture Elements ......................................... 7
5. Architecture Foundations ............................................ 8
6. Task Extensions ........................................................ 9
7. Task Specification ..................................................... 9
   7.1. CONCEPT for task type identification .......................... 10
   7.2. Task Context and Relationships ................................ 10
   7.3. Task Domain Data Handling ..................................... 10
8. Task Deadlines, Milestones and Time Planning ........................ 11
9. Task Scheduling and Assignment ...................................... 11
10. Status Reporting ..................................................... 12
   10.1. Improved granularity in status reporting information ....... 12
   10.2. Relating reason and comments to ATTENDEE status changes. 12
   10.3. Comments associated to reasons and status changes ........ 12
   10.4. Task Alerts and Notifications ................................ 13
   10.5. Automated Status Changes .................................... 14
11. New Parameter Values ................................................ 14
   11.1. Redefined VTODO Participant Status .......................... 14
12. New Properties ........................................................ 14
   12.1. Estimated Duration ............................................ 14
   12.2. Reason .......................................................... 15
   12.3. Sub-State ....................................................... 16
   12.4. Task Mode ....................................................... 17
13. Property Extensions and Clarifications ............................. 18
   13.1. Redefined STATUS Property .................................... 19
14. New Components ....................................................... 19
   14.1. Status Component .............................................. 19
15. CalDAV Support for Task Mode ....................................... 20
   15.1. CALDAV: supported-task-mode-set Property .................. 21
16. Security Considerations ............................................. 22
17. IANA Considerations ................................................ 22
   17.1. Initialization of the Status registry ........................ 22
   17.2. Update of the Status registry ................................ 22
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 "VTTODO" 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.
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 a new status component that can be used to group related information about the status. This might include information on why (REASON) and when (DTSTAMP) a status has changed. In addition new status values are specified to provide for task suspension, failure and preparation.

10.2. Relating reason and comments to ATTENDEE status changes.

The [RFC9073] PARTICIPANT component can be used to provide additional information about why an ATTENDEE participation status has changed. The COMMENT property can also be used to include additional human readable information about why the associated STATUS or ATTENDEE property changed.

```
BEGIN:VSTATUS
STATUS:FAILED
REASON:http://example.com/reason/delivery-failed
SUBSTATE:ERROR
DTSTAMP:20130212T120000Z
COMMENT:Breakdown
END:VSTATUS

ATTENDEE;PARTSTAT=FAILED:mailto:xxx@example.com
...
```

```
BEGIN:PARTICIPANT
CALENDAR-ADDRESS:mailto:xxx@example.com
DTSTAMP:20130226T1104510Z
REASON:http://example.com/reason/van-break-down
COMMENT:Puncture
END:PARTICIPANT
```

10.3. Comments associated to reasons and status changes

Multiple comments and reasons may have the same status. As situations change further VSTATUS components can be added to provide additional information..

```
```
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 Parameter Values

11.1. Redefined VTODO Participant Status

Participant status parameter type values are defined in Section 3.2.12 of [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\`

12. New Properties

12.1. 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 "VTTODO" 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

12.2.  Reason

Property name  REASON

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

Value Type  URI

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

Conformance  This property can be specified in "VSTATUS" and PARTICIPANT calendar components.

Format Definition  This property is defined by the following notation:

reason = "REASON" reasonparam ":" uri CRLF

reasonparam = *(";" other-param)

Description  This property 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 name-space of the authority defining the task. Common reason codes are IANA registered and do not have a name-space prefix.

Example

REASON:http://example.com/reason/delivered-on-time

REASON:out-of-office
12.3. Sub-State

Property name  SUBSTATE

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

Value Type  TEXT

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

Conformance  This property can be specified in a "VSTATUS" calendar component.

Format Definition  This property is defined by the following notation:

```
substate          = "SUBSTATE" substateparam ":" substatevalue CRLF
substateparam     = *(";" other-param)
substatevalue     = ("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)
/ iana-token) ; Other IANA-registered type
```

Description  The sub-state property 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
BEGIN:VSTATUS
STATUS:FAILED
REASON:http://example.com/reason/no-one-home
SUBSTATE:ERROR
END:VSTATUS

BEGIN:VSTATUS
STATUS:IN-PROCESS
REASON:http://example.com/reason/paint-drying
SUBSTATE:SUSPENDED
END:VSTATUS

12.4. 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:

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

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

\[
\text{taskmodeparam} = *(";" other-param)
\]

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 subsections.

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.

13. Property Extensions and Clarifications
13.1. Redefined STATUS Property

The Status property is defined in Section 3.8.1.11 of [RFC5545]. This specification extends that property to include 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:

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:FAILED

14. New Components

14.1. Status Component

Component Name VSTATUS

Purpose This component allows information to be associated with a status, for example comments and date stamps.

Conformance This component can be specified multiple times in any appropriate calendar component.

Description This component provides a way for multiple date-stamped statuses to be associated with a component such as a task or an event.
This component may also be added to the [RFC9073] PARTICIPANT component to allow participants in a task to specify their own status.

Format Definition  This component is defined by the following notation:

```
statusc = "BEGIN" ":" "VSTATUS" CRLF
  statusprop
"END" ":" "VSTATUS" CRLF
```

```c
statusprop = *( ;
; The following is REQUIRED,
; but MUST NOT occur more than once.
; status />
; ; The following are OPTIONAL,
; but MUST NOT occur more than once.
; ; description / dtstamp / reason / substate / summary
; ; The following are OPTIONAL,
; ; and MAY occur more than once.
; ; comment / styleddescription / iana-prop
; )
```

Examples

```
BEGIN:VSTATUS
STATUS:COMPLETED
REASON: http://example.com/reason/delivered-on-time
DTSTAMP:20220212T120000Z
END:VSTATUS
```

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

```xml
<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>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCELLED</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>COMPLETED</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>CONFIRMED</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>DRAFT</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>FINAL</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
<tr>
<td>TENTATIVE</td>
<td>Current</td>
<td>Section 3.8.1.11 of [RFC5545]</td>
</tr>
</tbody>
</table>

Table 1: Initial Status Value Registry

17.2. Update of the Status registry

This specification further updates the Status registry with additional values defined in this document.
17.3. Sub-State value registry

The following table has been used to initialize the Sub-State registry.

<table>
<thead>
<tr>
<th>Substate</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Current</td>
<td>This Spec, Section 12.3</td>
</tr>
<tr>
<td>ERROR</td>
<td>Current</td>
<td>This Spec, Section 12.3</td>
</tr>
<tr>
<td>SUSPENDED</td>
<td>Current</td>
<td>This Spec, Section 12.3</td>
</tr>
</tbody>
</table>

Table 3: 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 12.4</td>
</tr>
<tr>
<td>AUTOMATIC-FAILURE</td>
<td>Current</td>
<td>This Spec, Section 12.4</td>
</tr>
<tr>
<td>CLIENT</td>
<td>Current</td>
<td>This Spec, Section 12.4</td>
</tr>
<tr>
<td>SERVER</td>
<td>Current</td>
<td>This Spec, Section 12.4</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.

+--------+---------+-------------------------+
| Value   | Status  | Reference               |
+--------+---------+-------------------------+
| FAILED  | Current | This Spec, Section 11.1 |
+--------+---------+-------------------------+

Table 5: Participation Statuses Registry

17.6. Properties registry

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

+====================+=========+=========================+
| Property            | Status  | Reference               |
+====================+=========+=========================+
| ESTIMATED_DURATION  | Current | This Spec, Section 12.1 |
+---------------------+---------+-------------------------+
| REASON              | Current | This Spec, Section 12.2 |
+---------------------+---------+-------------------------+
| SUBSTATE            | Current | This Spec, Section 12.3 |
+---------------------+---------+-------------------------+
| STATUS              | Current | This Spec, Section 13.1 |
+---------------------+---------+-------------------------+
| TASK-MODE           | Current | This Spec, Section 12.4 |
+---------------------+---------+-------------------------+

Table 6: Updated Properties Registry

18. Normative References


Apthorp & Douglass Expires 22 September 2022 [Page 24]


19. Informative References


Appendix A. Examples of Task State Lifecycle

A.1. Simple Case Status Change

+===+==============+==============+===========================+  
|   | STATUS       | PARTSTAT     | Action                    |  
+---+--------------+--------------+---------------------------+  
| 1 | -            | -            | Organizer draft           |  
| 2 | NEEDS-ACTION | NEEDS-ACTION | Organizer sends iTIP      |  
|   |              |              | request                   |  
| 3 | NEEDS-ACTION | ACCEPTED     | Attendee reply            |  
| 4 | PENDING      | ACCEPTED     | Task accepted but waiting |  
|   |              |              | on some "trigger" to      |  
|   |              |              | start (e.g. another task  |  
|   |              |              | has to finish first)      |  
| 5 | IN-PROCESS   | IN-PROCESS   | Attendee reply now        |  
|   |              |              | working on the task       |  
| 6 | IN-PROCESS   | COMPLETED    | Attendee reply completed  |  
| 7 | COMPLETED    | COMPLETED    | Organizer changes overall |  
|   |              |              | state                     |  

Table 7: 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>Organizer</td>
<td>-</td>
<td>-</td>
<td>Organizer draft.</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>NEEDS-ACTION</td>
<td>NEEDS-ACTION</td>
<td>Organizer sends iTIP request.</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>ACCEPTED</td>
<td>NEEDS-ACTION</td>
<td>Attendee 1 reply.</td>
</tr>
<tr>
<td>NEEDS-ACTION</td>
<td>ACCEPTED</td>
<td>ACCEPTED</td>
<td>Attendee 2 reply.</td>
</tr>
<tr>
<td>PENDING</td>
<td>ACCEPTED</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>IN-PROCESS</td>
<td>ACCEPTED</td>
<td>IN-PROCESS</td>
<td>Attendee 2 reply now working on the task.</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>IN-PROCESS</td>
<td>IN-PROCESS</td>
<td>Attendee 1 reply now working on the task.</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>COMPLETED</td>
<td>IN-PROCESS</td>
<td>Attendee 1 reply Completed (overall status still IN-PROCESS).</td>
</tr>
<tr>
<td>IN-PROCESS</td>
<td>COMPLETED</td>
<td>COMPLETED</td>
<td>Attendee 2 reply Completed</td>
</tr>
<tr>
<td>COMPLETED</td>
<td>COMPLETED</td>
<td>COMPLETED</td>
<td>Organizer changes</td>
</tr>
</tbody>
</table>
Table 8: Example for multiple Attendees

Note: The logic for determining the status change to the VTODO is determined by the task organizer based on the ATTENDEE status and other business logic.

A.3. Example of Failure

Example of status changes for a task that fails.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>PARSTAT</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 9: Example of Failure

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
Fritz-Erler-Str. 5
Bonn
Germany
Email: adrian.apthorp@dhl.com
Michael Douglass  
Bedework Commercial Services  
226 3rd Street  
Troy, NY  
United States of America  
Email: mdouglass@bedework.com
Support for Series in iCalendar
draft-ietf-calext-icalendar-series-03

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 14 October 2021.

Copyright Notice

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

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

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

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

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

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

3. Terms and definitions

4. Overrides and iCalendar recurrences

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

The [RFC5545] specification allows for overrides. These are handled by supplying a complete replacement for the instance with a RECURRENCE-ID property matching that of the instance being overridden. This may change any of the properties (except the UID) - including start, end or duration.

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

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

There are approaches being considered to mitigate some of these issues which mostly involve only storing changes. This can help if the changes are minor but heavily modified instances are still a problem.

4.1. Changing the master start or the recurrence rules

This can lead to some very difficult problems to resolve. In the case of a heavily modified meeting it may be difficult to impossible to determine which override applies to the newly modified event.
For example, a weekly book-reading is moved from Monday to Friday. There are weeks of scheduled events in the future. Do we move them all forward to the next instance or skip one and move them back? If it becomes bi-weekly rather than weekly do we drop every other or just space them out more?

While these problems are not totally resolved by a series approach, they become more tractable.

4.2. Splitting recurrences

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

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

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

5. Series

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

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

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

Each instance and master is identified as a member of the full series by the SERIES-UID property. The value of this property is the same in all members of the series even when splits have occurred.
As instances are created a LAST-SERIES-ID property is added or updated in the master to indicate which instance was last created. When there are SXDATE properties this property value may represent an instance which cannot be created. It merely represents the latest calculated date.

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

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

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

5.1. Modifying series patterns and splitting

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

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

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

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

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

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

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

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

5.3. The series instances

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

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

6. Redefined Relation Type Value

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

Format Definition

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

Figure 1

Description

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

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

RELTYPE=PARENT See [RFC5545].
RELTYPE=CHILD See [RFC5545].
RELTYPE=SIBLING See [RFC5545].
RELTYPE=DEPENDS-ON Indicates that the current calendar component
depends on the referenced calendar component in some manner. For example a task may be blocked waiting on the other, referenced, task.

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

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

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

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

```
  =========
  |  Task-A  |-->
  =========
     V
  =========
   |  Task-B  |
  =========
```

Figure 2: Finish to start relationship

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

```
  =========
  |  Task-A  |-->
  =========
  =========
   |  Task-B  |<--
  =========
```

Figure 3: Finish to finish relationship
RELTYPE=STARTTOFINISH

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

```
+--|  Task-A  |
    ============
    |
    ============
    |
    Task-B |<--
    ============
```

Figure 4: Start to finish relationship

RELTYPE=STARTTOSTART

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

```
+--|  Task-A  |
    ============
    |
    ============
    +->|  Task-B  |
    ============
```

Figure 5: Start to start relationship

7. New Property Parameters

7.1. Split

Parameter name SPLIT

Purpose To indicate a series has been split.

Format Definition

This parameter is defined by the following notation:

```
splitparam = "SPLIT" "="
("YES" ; The series is split
/ "NO" ; The series is not split (default)
/ x-name ; Experimental reference type
/ iana-token) ; Other IANA registered type
```
Figure 6

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

7.2. Lookahead count

Parameter name  LOOKAHEAD-COUNT

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

Format Definition

This parameter is defined by the following notation:

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

Figure 7

Description

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

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

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

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

7.3. Lookahead period

Parameter name  LOOKAHEAD-PERIOD

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

Format Definition

This parameter is defined by the following notation:
lookahead-periodparam = "LOOKAHEAD-PERIOD" "="
DQUOTE dur-value DQUOTE

Figure 8

Description

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

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

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

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

The value is a quoted duration.

8. New Properties

8.1. General

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

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

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

8.2. Generating Series members

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

Elements of the rule The UNTIL or COUNT parts of the rule define when the series terminates. Thus a COUNT=100 specifies a maximum of 100 series members.
Lookahead count  This specifies how many series members can exist from the current date/time. Thus a LOOKAHEAD-COUNT=4 means a maximum of 4 generated instances.

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

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

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

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

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

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

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

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

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

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

8.3. Series UID

Property name SERIES-UID

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

Value Type TEXT

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

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

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

Format Definition

This property is defined by the following notation:

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

seruidparam = *(";" other-param)

Figure 9

EXAMPLE

The following is an example of this property:

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

Figure 10

8.4. Series-exception-date

Property name SXDATE

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

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

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

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

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

Format Definition

This property is defined by the following notation:

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

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

Figure 11

EXAMPLE

The following is an example of this property:

Douglass                 Expires 14 October 2021               
                            [Page 14]
Figure 12

8.5. Series-date

Property name SDATE

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

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

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

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

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

Format Definition

This property is defined by the following notation:
sdate = "SDATE" sdtparam \:" sdtval *("," sdtval) CRLF
sdtparam = *( ;
; The following are OPTIONAL,
; but MUST NOT occur more than once.
; ;
; ";" "VALUE" "=" ("DATE-TIME" / "DATE" / "PERIOD"))
; ;
; The following is OPTIONAL,
; ; and MAY occur more than once.
; ;
; ";" other-param)
; )

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

EXAMPLE

The following are examples of this property:

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

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

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

8.6. Series-id

Property name SERIES-ID

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

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

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

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

Description

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

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

Figure 15

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

Figure 16

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

Figure 17

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

Figure 18

Format Definition

This property is defined by the following notation:
serid   = "SERIES-ID" sidparam ":" sidval CRLF

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

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

Figure 19

EXAMPLE

The following are examples of this property:

SERIES-ID;VALUE=DATE:19960401

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

Figure 20

8.7. Last series ID

Property name LAST-SERIES-ID

Purpose

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

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

Value type DATE or DATE_TIME (the default). This has the same
requirements as SERIES-ID.
Property Parameters  IANA, non-standard, value data type and time zone identifier parameters can be specified on this property.

Conformance  This property MAY be specified in any iCalendar component.

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

Format Definition

This property is defined by the following notation:
last-series-i = "LAST-SERIES-ID" lastseriesidparam / 
  ( 
    ";" "VALUE" "=" "TEXT"
    ":" text 
  ) 
  ( 
    ";" "VALUE" "=" "REFERENCE"
    ":" text 
  ) 
  ( 
    ";" "VALUE" "=" "URI"
    ":" uri 
  )
CRLF

lastseriesidparam = *( 
  ; the following is MANDATORY 
  ; and MAY occur more than once 
  (";" relparam) / 
  ; the following are MANDATORY 
  ; but MUST NOT occur more than once 
  (";" fmttypeparam) / 
  (";" labelparam) / 
  ; labelparam is defined in ... 
  ; the following is OPTIONAL 
  ; and MAY occur more than once 
  (";" xparam) 
  )

Figure 21

EXAMPLE

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

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

Figure 22
8.8. Series Rule

Property name SRULE

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

Value Type RECUR

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

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

Description

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

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

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

The "DTSTART" property specified within the master iCalendar object defines the first instance of the recurrence. In most cases, a "DTSTART" property of DATE-TIME value type used with a series rule, should be specified as a date with local time and time zone reference to make sure all the recurrence instances start at the same local time regardless of time zone changes.
If the duration of the series component is specified with the "DTEND" or "DUE" property, then the same exact duration will apply to all the members of the generated series. Else, if the duration of the series master component is specified with the "DURATION" property, then the same nominal duration will apply to all the members of the generated series and the exact duration of each instance will depend on its specific start time. For example, series instances of a nominal duration of one day will have an exact duration of more or less than 24 hours on a day where a time zone shift occurs. The duration of a specific instance may be modified in an exception component or simply by using an "SDATE" property of PERIOD value type.

Format Definition

This property is defined by the following notation:

```
srule = "SRULE" srulparam ":" recur CRLF
sruleparam = *( ; the following are OPTIONAL
  ; but MUST NOT occur more than once
  (";" lookahead-countparam) /
  (";" lookahead-periodparam) /
  ; the following is OPTIONAL
  ; and MAY occur more than once
  (";" xparam)
)
```

Figure 23

EXAMPLE

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

9. Redefined RELATED-TO Property

9.1. RELATED-TO

Property name RELATED-TO

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

Conformance  This property MAY be specified in any iCalendar component.

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

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

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

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

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

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

Format Definition  This property is defined by the following notation:
related = "RELATED-TO" relparam ( ":" text ) /  
  ( ":" "VALUE" ":" "UID"  
    ":" uid  
  )  
  ( ":" "VALUE" ":" "URI"  
    ":" uri  
  )  
CRLF  
  relparam = *(
    ; The following are OPTIONAL,  
    ; but MUST NOT occur more than once.  
    ;(";" reltypeparam) /  
    (";" gapparam) /  
    ; The following is OPTIONAL,  
    ; and MAY occur more than once.  
    ;(";" other-param)  
  )  

Figure 24

EXAMPLE

The following are examples of this property.

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

Figure 25

10. Backwards compatibility

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

For such clients the series appears to be an unconnected set of components. They all have their own unique UIDS. If the client updates an instance this should be identical in effect to an update carried out by a client aware of the new properties.
Updates MUST preserve the SERIES-ID, LAST-SERIES-ID, SRULE, SDATE and SXDATE properties. A client which does not do so is in violation of [RFC5545].

TODO - More text needed here... == CalDAV extensions

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

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

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

11. IANA Considerations

11.1. iCalendar Property Registrations

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

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

Table 1

11.2. iCalendar Property Parameter Registrations

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

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

<table>
<thead>
<tr>
<th>Relationship Type</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIES-ID</td>
<td>Current</td>
<td>Section 5</td>
</tr>
</tbody>
</table>

Table 3

12. Normative References


13. Informative References


Author's Address

Douglass

Expires 14 October 2021
Michael Douglass

Email: mikeadouglass@gmail.com
JSCalendar: Converting from and to iCalendar
draft-ietf-calext-jscalendar-icalendar-06

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 17 July 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised 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 ................................. 6  
3. Translating iCalendar components to JSCalendar .......... 7  
   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 ...................................... 16  
   4.7. COMMENT ................................... 17  
   4.8. COMPLETED ................................... 17  
   4.9. CONCEPT .................................... 17  
   4.10. CONFERENCE ................................ 18  
   4.11. CONTACT .................................... 18  
   4.12. CREATED .................................... 18  
   4.13. DESCRIPTION ................................ 19  
   4.14. DTEND, DTSTART, DUE, DURATION .................. 20  
   4.15. ESTIMATED-DURATION ............................. 23  
   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 ................................... 24  
   4.24. PERCENT-COMPLETE ............................... 25  
   4.25. PRIORITY .................................... 25  
   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
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 [RFC8984] data format is used to represent calendar data, and is meant as an alternative to the widely deployed iCalendar [RFC5545] data format.

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

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

1.3. Scope and caveats

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

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

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

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

* 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) [RFC5545]

* iCalendar Transport-Independent Interoperability Protocol (iTIP)
  [RFC5546]

* New Properties for iCalendar [RFC7986]

* Event Publishing Extensions to iCalendar [RFC9073]

* Support for iCalendar Relationships
  [draft-ietf-calext-ical-relations]

* "VALARM" Extensions for iCalendar [RFC9074]

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 [RFC8259], 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:AUDIO
ATTACH;FMTTYPE=audio/basic:ftp://example.com/pub/sounds/bell-01.aud
END:VALARM
BEGIN:VALARM
TRIGGER:-PT30M
REPEAT:2
DURATION:PT15M
ACTION:DISPLAY
DESCRIPTION:Breakfast meeting with executive\n team at 8:30 AM EST.
END:VALARM
BEGIN:VALARM
TRIGGER;RELATED=END:-P2D
ACTION:EMAIL
ATTENDEE:mailto:john_doe@example.com
SUMMARY:*** REMINDER: SEND AGENDA FOR WEEKLY STAFF MEETING ***
DESCRIPTION:A draft agenda needs to be sent out to the attendees to the weekly managers meeting (MGR-LIST). Attached is a
maps to

{
   "@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 "Group".

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

maps to

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

Note that a single Event or Task MAY be converted without a surrounding Group 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 [RFC9073] 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": "Event",
  ...
  "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": "Event",
  ...
  "timezones": {
    "/Example/Somewhere": {
      "@type": "TimeZone",
      "tzId": "Example/Somewhere",
      ...
    },
    "/Example/Somewhere-else": {
      "@type": "TimeZone",
      "tzId": "Example/Somewhere-else",
      ...
    }
  }
}

3.6. VTODO

A [RFC5545] VTODO component is mapped to a JSCalendar object with a type of "Task".
BEGIN: VTODO
...
END: VTODO
/maps to
{
  "@type": "Task",
  ...
}
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:
ATTENDEE:mailto:user01@example.org

maps to

{
  ...
  "participants": {
    "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
      "@type": "Participant",
      "sendTo": {
        "imip": "mailto: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: "20101010T101010Z"
... 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": "Event",
  ...
  "created": "1996-03-29T13:30Z",
  ...
}

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": "Event",
   ...
   "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. 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": "Event",
  ...
  "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": "Event",
    ...
    "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": "Event",
    ...
    "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:

```plaintext
... 
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 [RFC9073] "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 "imp" 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": "Task",
  "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": "Task",
   "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:VTTODO
...
ATTENDEE:mailto:douglm@example.org
END:VTTODO
BEGIN:VTTODO
...
RECURRENCE-ID:20200523T120000
...
ATTENDEE:mailto:douglm@example.org
PERCENT-COMPLETE:39
END:VTTODO
END: VCALENDAR

maps to

{
   "@type": "Task",
   "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 Group object with multiple Event and/or Task object the [RFC5545] VCALENDAR PRODID is mapped to a JSCalendar "prodid" property in the group.

When mapping to a single Event and/or Task 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": "Event",
  "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": "Event",
  "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
    }]
  }], ...
...

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

BYMONT MONTH 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.

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 there is no time part append a 0 time and reformat as a jscalendar local date/time.

Some examples:

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

...  
"recurrenceRules" : [{  
   "@type" : "RecurrenceRule",  
   "frequency" : "daily",  
   "count" : 10  
 }],  
...
RRULE:FREQ=YEARLY;UNTIL=20220512T140000Z;BYMONTH=1;BYDAY=SU,MO,TU,WE,TH,FR,SA

maps to

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

maps to ...

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

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;ewogICAigIAICAY29udGV4dGV4dCI6IC..."
  }
}
...
```

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

{ }
"prodId": "//example.org//EN",
"entries": [
  {
    "links": {
      "1": {
        "@type": "Link",
        "rel": "enclosure",
        "contentType": "text/plain",
        "href": "http://example.org/doc1.txt"
      },
      "2": {
        "@type": "Link",
        "rel": "enclosure",
        "contentType": "text/plain",
        "href": "http://example.org/doc2.txt"
      },
      "3": {
        "@type": "Link",
        "rel": "enclosure",
        "contentType": "text/plain",
        "href": "http://example.org/doc3.txt"
      }
    },
    "created": "2020-05-23T17:04:50Z",
    "start": "2020-05-22T12:00:00",
    "timeZone": "America/New_York",
    "duration": "PT1H",
    "title": "recurring daily 8 times",
    "uid": "6252D6C40A8308BFE25BBDErecur-1",
    "recurrenceRules": [
      {
        "@type": "RecurrenceRule",
        "frequency": "daily",
        "count": 8
      }
    ],
    "recurrenceOverrides": {
      "2020-05-23T12:00:00": {
        "recurrenceId": "2020-05-23T12:00:00",
        "links/d4a618d4-929c-4c81-ae5b-322afe407a00": null,
        "links/fb75b76a-a159-4a86-bd3d-7ace6b39c6c3": {
          "@type": "Link",
          "rel": "enclosure",
          "contentType": "text/plain",
          "href": "http://example.org/doc4.txt"
        }
      },
      "2020-05-24T12:00:00": {
        "recurrenceId": "2020-05-24T12:00:00",
        "links/d4a618d4-929c-4c81-ae5b-322afe407a00": null,
        "links/fb75b76a-a159-4a86-bd3d-7ace6b39c6c3": {
          "@type": "Link",
          "rel": "enclosure",
          "contentType": "text/plain",
          "href": "http://example.org/doc4.txt"
        }
      }
    }
  }
]
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": "Group",
  "prodId": "//Example.org//Example V3.13.2//EN",
  "entries": [
    {
      "@type": "Event",
      "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/6c5e72e-3413-487c-ae14-fb318a90db43": null,
            "links/44087e9a-132c-4a5d-b25d-4ce580edb004": 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.

BEGIN:VCALENDAR
PRODID://Example.org//Example V3.13.2//EN
VERSION:2.0
BEGIN:VEVENT
CREATED:20200704T035515Z
DURATION:PT1H
DTSTAMP:20200704T035706Z
DTSTART;TZID=America/New_York:20200622T120000
LAST-MODIFIED:20200704T035706Z
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
END:VCALENDAR
RRULE:FREQ=DAILY;COUNT=8
END:VEVENT
BEGIN:VEVENT
RECURRENCE-ID;TZID=America/New_York:20200523T120000
ATTENDEE:mailto:douglm@example.org
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2;DIR="http://example.org/vcards/vbede.vcf":mailto:vbede@example.org
CREATED:20200704T035515Z
DURATION:PT1H
DTSTAMP:20200704T035706Z
DTSTART;TZID=America/New_York:20200523T120000
LAST-MODIFIED:20200704T035706Z
ORGANIZER:mailto:douglm@example.org
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
BEGIN:VEVENT
RECURRENCE-ID;TZID=America/New_York:20200524T120000
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2:mailto:user01@example.org
ATTENDEE;RSVP=TRUE;SCHEDULE-STATUS=1.2:mailto:vbede@example.org
CREATED:20200704T035515Z
DURATION:PT1H
DTSTAMP:20200704T035706Z
DTSTART;TZID=America/New_York:20200524T120000
LAST-MODIFIED:20200704T035706Z
ORGANIZER:mailto:douglm@example.org
SUMMARY:recurring daily 8 times
UID:6252D6C40A8308BFE25BBEFrecur1-1
END:VEVENT
END:VCALENDAR

translates to

{
  "@type": "Group",
  "prodId": "//Example.org//Example V3.13.2//EN",
  "entries": [
    {
      "@type": "Event",
      "created": "2020-07-04T03:57:06Z",
      "start": "2020-05-22T12:00:00",
      "timeZone": "America/New_York",
      "duration": "PT1H",
      "title": "recurring daily 8 times",
      "uid": "6252D6C40A8308BFE25BBEFrecur1-1",
      "recurrenceRules": [
        {
          "@type": "RecurrenceRule",

"frequency": "daily",
"count": 8
}
],
"recurrenceOverrides": {
  "2020-05-23T12:00:00": {
    "participants": {
      "be450b70-9bf7-4f6e-8f65-971ede566ce3": {
        "@type": "Participant",
        "sendTo": {
          "imip": "mailto:douglm@example.org"
        },
        "roles": {
          "attendee": true,
          "owner": true
        }
      },
      "a539dfe3-4463-4f28-b9de-17d3a0e99faf": {
        "@type": "Participant",
        "sendTo": {
          "imip": "mailto:vbede@example.org"
        },
        "expectReply": true,
        "links": {
          "1": {
            "@type": "Link",
            "href": "http://example.org/vcards/vbede.vcf",
            "rel": "alternate"
          }
        },
        "roles": {
          "attendee": true
        },
        "scheduleStatus": "1.2"
      }
    }
  },
  "replyTo": {
    "imip": "mailto:douglm@example.org"
  }
},
  "2020-05-24T12:00:00": {
    "participants": {
      "daee4cf6-6f6a-4ce3-9f4d-6bd884650d3d": {
        "@type": "Participant",
        "sendTo": {
          "imip": "mailto:user01@example.org"
        },
        "expectReply": true,
"roles": {
    "attendee": true
},
"scheduleStatus": "1.2"
},
"a6de6de3-271f-4679-9241-1b3bca6b602d": {
    
    "@type": "Participant",
    "sendTo": {
        "imip": "mailto:vbede@example.org"
    },
    "expectReply": true,
    "roles": {
        "attendee": true
    },
    "scheduleStatus": "1.2"
},
"aaa8483b-b18b-4dbd-b218-77d8db027d35": {
    "@type": "Participant",
    "sendTo": {
        "imip": "mailto:douglm@example.org"
    },
    "roles": {
        "owner": true
    }
},
"replyTo": {
    "imip": "mailto:douglm@example.org"
}
]

7. Translating JSCalendar objects to iCalendar

This section lists the JSCalendar objects that map to [RFC5545] components.

7.1. Event

A JSCalendar object with a type of "Event" 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.
maps to

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

When converting multiple Event or Task objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the Group "prodid" or generated.

7.2. Group

A JSCalendar object with a type of "Group" is mapped on to a [RFC5545] VCALENDAR component.
7.3. Task

A JSCalendar object with a type of "Task" 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.
When converting multiple Event or Task objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the Group "prodid" or generated.

8. Translating JSCalendar properties to iCalendar

This section is an alphabetic list of all JSCalendar top-level properties that map to [RFC5545] iCalendar.

8.1. alerts

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

8.1.1. 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

8.1.2. 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

8.1.3. todo

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

8.2. categories

Each member of the JSCalendar "categories" property maps on to a [RFC9073] 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
...
8.3. 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:

```json
...
"created": "2021-03-15T13:30"0Z"
...
```

maps to

```text
...
CREATED:20210315T133000Z
...
```

8.4. duration

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

For example:

```json
...
"duration": "PT1H"
...
```

maps to

```text
...
DURATION:PT1H
...
```

8.5. estimatedDuration

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

For example:
... "estimatedDuration": "PT18H"
...

maps to

... ESTIMATED-DURATION:PT18H
...

8.6. 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
...

8.7. locations

JSCalendar locations map to [RFC9073]VLOCATION components.

Additionally, for backwards compatibility, one location should be mapped on to a [RFC5545] LOCATION property.
8.7.1. coordinates

TODO. Need to decide if to use GEO or define new GEO-URI.

8.7.2. description

The "description" property maps to the DESCRIPTON property value of the VLOCATION.

8.7.3. links

TODO. First need to define top-level links property mapping.

8.7.4. locationTypes

The keys of the "locationTypes" property map to the LOCATION-TYPE property value of the VLOCATION. The keys MUST be separated by the COMMA character (U+002c) and SHOULD sort in ascending alphabetical order.

8.7.5. name

The "name" property maps to the NAME property value of the VLOCATION.

8.7.6. relativeTo

The "relativeTo" property maps to the RELATED-TO property value of the VLOCATION. TODO need updated definition of the RELATED-TO draft.

8.7.7. timeZone

The "timeZone" property maps to the TZID property value of the VLOCATION.

If the TimeZoneId value matches a name from the IANA Time Zone Database [TZDB] then this value MUST be set in the TZID property.

If the TimeZoneId identifies a custom TimeZone in the JSCalendar object, then the TZID property value MUST be set to the "tzId" property value of the custom TimeZone object, and its related VTIMEZONE added to the VCALENDAR component that encloses the VLOCATION.
8.7.8. uid

The "uid" property maps to the UID property value of the VLOCATION. The identifier of the Location object in the enclosing "locations" property maps to the JMAP-ID property parameter. The parameter MAY be omitted if the identifier of the Location matches the "uid" value.

8.8. 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 [RFC9073]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 [RFC9073]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 [RFC9073]PARTICIPANT component which provides a better mapping.

A more complete mapping may be achieved by creating a [RFC9073]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?

8.9. 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.


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

maps to

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

When converting multiple Event or Task objects the surrounding [RFC5545] VCALENDAR object must have a [RFC5545] PRODID set from either the Group "prodid" or generated.

9. Security Considerations

The same security considerations as for [RFC8984] 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


12.2. Informative References

[TZDB] IANA, "Time Zone Database",

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 defines a data model and JSON representation of contact card information that can be used for data storage and exchange in address book or directory applications. It aims to be an alternative to the vCard data format and to be unambiguous, extendable and simple to process. In contrast to the JSON-based jCard format, it is not a direct mapping from the vCard data model and expands semantics where appropriate.

Status of This Memo

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

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

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

This Internet-Draft will expire on 13 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
# Table of Contents

1. Introduction .............................................. 3
   1.1. Relation to the xCard and jCard formats ............ 4
   1.2. Terminology ......................................... 4
   1.3. Vendor-specific Property Extensions and Values ...... 4
   1.4. Type Signatures ...................................... 5
   1.5. Data types ........................................... 5
       1.5.1. Context ........................................ 5
       1.5.2. Id ............................................... 6
       1.5.3. PatchObject .................................. 6
       1.5.4. Preference .................................... 7
       1.5.5. UnsignedInt .................................. 7
       1.5.6. UTCDateTime ................................. 8

2. Card ......................................................... 8
   2.1. Metadata properties .................................. 8
       2.1.1. @type ........................................ 8
       2.1.2. uid .......................................... 8
       2.1.3. prodId ....................................... 8
       2.1.4. created ..................................... 8
       2.1.5. updated ..................................... 9
       2.1.6. kind ......................................... 9
       2.1.7. relatedTo .................................... 9
       2.1.8. language .................................... 10

2.2. Name and Organization properties ......................... 10
   2.2.1. name .......................................... 10
   2.2.2. fullName ....................................... 11
   2.2.3. nickNames ..................................... 11
   2.2.4. organizations ................................ 12
   2.2.5. titles ......................................... 12
   2.2.6. speakToAs ..................................... 12

2.3. Contact and Resource properties .......................... 13
   2.3.1. emails ......................................... 13
   2.3.2. phones ......................................... 14
   2.3.3. online .......................................... 15
   2.3.4. scheduling ..................................... 16
   2.3.5. photos ......................................... 17
   2.3.6. preferredContactMethod ......................... 17
   2.3.7. preferredContactLanguages ...................... 17
1. Introduction

This document defines a data model for contact card data normally used in address book or directory applications and services. It aims to be an alternative to the vCard data format [RFC6350] and to provide a JSON-based standard representation of contact card data.

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

* Most of the initial set of attributes should be taken from the vCard data format [RFC6350] and extensions ([RFC6473], [RFC6474], [RFC6715], [RFC6869], [RFC8605]). The specification should add new attributes or value types, or not support existing ones, where appropriate. Conversion between the data formats need not fully preserve semantic meaning.

* The attributes of the cards data represented must be described as a simple key-value pair, reducing complexity of its representation.

* The data model should avoid all ambiguities and make it difficult to make mistakes during implementation.
* Extensions, such as new properties and components, MUST NOT lead
to requiring an update to this document.

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

1.1. Relation to the xCard and jCard formats

The xCard [RFC6351] and jCard [RFC7095] specifications define
alternative representations for vCard data, in XML and JSON format
respectively. Both explicitly aim to not change the underlying data
model. Accordingly, they are regarded as equal to vCard in the
context of this document.

1.2. Terminology

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.

1.3. Vendor-specific Property Extensions and Values

Vendors MAY add additional properties to the contact object to
support their custom features. To avoid conflict, the names of these
properties MUST be prefixed by a domain name controlled by the vendor
followed by a colon, e.g., "example.com:customprop". If the value is
a new JSContact object, it either MUST include an "@type" property,
or it MUST explicitly be specified to not require a type designator.
The type name MUST be prefixed with a domain name controlled by the
vendor.

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

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

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

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

* String - The JSON string type.

* Number - The JSON number type.

* Boolean - The JSON boolean type.

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

* A[] - An array of values of type A.

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

1.5. Data types

In addition to the standard JSON data types, a couple of additional data types are common to the definitions of JSContact objects and properties.

1.5.1. Context

Contact information typically is associated with a context in which it should be used. For example, someone might have distinct phone numbers for work and private contexts. The Context data type enumerates common contexts.

Common context values are:

* private: The contact information may be used to contact the card holder in a private context.

* work: The contact information may be used to contact the card holder in a professional context.

Additional allowed values may be defined in the properties or data types that make use of the Context data type, registered in a future RFC, or a vendor-specific value.
1.5.2. Id

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

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

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

1.5.3. PatchObject

A PatchObject is of type String[*], and represents an unordered set of patches on a JSON object. Each key is a path represented in a subset of JSON pointer format [RFC6901]. The paths have an implicit leading /, so each key is prefixed with / before applying the JSON pointer evaluation algorithm.

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

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

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

3. There MUST NOT be two patches in the PatchObject where the pointer of one is the prefix of the pointer of the other, e.g., addresses/1/city and addresses.
4. The value for the patch MUST be valid for the property being set (of the correct type and obeying any other applicable restrictions), or if null the property MUST be optional.

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

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

* If non-null, set the value given as the value for this property (this may be a replacement or addition to the object being patched).

A PatchObject does not define its own @type property. Instead, a @type property in a patch MUST be handled as any other patched property value.

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

1.5.4. Preference

This data type allows to define a preference order on same-typed contact information. For example, a card holder may have two email addresses and prefer to be contacted with one of them.

A preference value MUST be an integer number in the range 1 and 100. Lower values correspond to a higher level of preference, with 1 being most preferred. If no preference is set, then the contact information MUST be interpreted as being least preferred.

Note that the preference only is defined in relation to contact information of the same type. For example, the preference orders within emails and phone numbers are independent of each other. Also note that the _preferredContactMethod_ property allows to define a preferred contact method across method types.

1.5.5. UnsignedInt

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

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

For example, 2010-10-10T10:10:10.003Z is conformant, but 2010-10-10T10:10:10.000Z is invalid and is correctly encoded as 2010-10-10T10:10Z.

2. Card

MIME type: application/jscontact+json;type=card

A Card object stores information about a person, organization or company.

2.1. Metadata properties

2.1.1. @type

Type: String (mandatory).

Specifies the type of this object. This MUST be Card.

2.1.2. uid

Type: String (mandatory).

An identifier, used to associate the object as the same across different systems, addressbooks and views. [RFC4122] describes a range of established algorithms to generate universally unique identifiers (UUID), and the random or pseudo-random version is recommended. For compatibility with [RFC6350] UIDs, implementations MUST accept both URI and free-form text.

2.1.3. prodId

Type: String (optional).

The identifier for the product that created the Card object.

2.1.4. created

Type: UTCDateTime (optional).
2.1.5.  updated

Type: UTCDateTime (optional).

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

2.1.6.  kind

Type: String (optional). The kind of the entity the Card represents.

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

* individual: a single person
* org: an organization
* location: a named location
* device: a device, such as appliances, computers, or network elements
* application: a software application

2.1.7.  relatedTo

Type: String[Relation] (optional).

Relates the object to other Card and CardGroup objects. This is represented as a map, where each key is the uid of the related Card or CardGroup and the value defines the relation. The Relation object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Relation.
* relation: String[Boolean] (optional, default: empty Object) Describes how the linked object is related to the linking object. The relation is defined as a set of relation types. If empty, the relationship between the two objects is unspecified. Keys in the set MUST be one of the RELATED property [RFC6350] type parameter values, or an IANA-registered value, or a vendor-specific value. The value for each key in the set MUST be true.
2.1.8.  language

Type: String (optional).

This defines the locale in which free-text property values can be assumed to be written in. The value MUST be a language tag as defined in [RFC5646]. Note that such values MAY be localized in the localizations Section 2.5.1 property.

2.2.  Name and Organization properties

2.2.1.  name

Type: Name (optional).

The name of the entity represented by this Card.

A Name object has the following properties

* @type: Name (mandatory). Specifies the type of this object. This MUST be Name.

* components: NameComponent[] (mandatory). The components making up the name. The component list MUST have at least one entry. Name components SHOULD be ordered such that their values joined by whitespace produce a valid full name of this entity. Doing so, implementations MAY ignore any components of type separator.

* locale: String (optional). The locale of the name. The value MUST be a language tag as defined [RFC5646].

A NameComponent object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be NameComponent.

* value: String (mandatory). The value of this name component.

* type: String (mandatory). The type of this name component. The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:

  - prefix. The value is a honorific title(s), e.g. "Mr", "Ms", "Dr".

  - given. The value is a given name, also known as "first name", "personal name".
- surname. The value is a surname, also known as "last name", "family name".
- middle. The value is a middle name, also known as "additional name".
- suffix. The value is a honorific suffix, e.g. "B.A.", "Esq.".
- separator. A formatting separator for two name components. The value property of the component includes the verbatim separator, for example a newline character.

* nth: UnsignedInt (optional, default: 1). Defines the rank of this name component to other name components of the same type. If set, the property value MUST be higher than or equal to 1.

For example, two name components of type surname may have their nth property value set to 1 and 2, respectively. In this case, the first name component defines the surname, and the second name component the secondary surname.

Note that this property value does not indicate the order in which to print name components of the same type. Some cultures print the secondary surname before the first surname, others the first before the second. Implementations SHOULD inspect the locale property of the Name object to determine the appropriate formatting. They MAY print name components in order of appearance in the components property of the Name object.

2.2.2. FullName

Type: String (optional).

The full name (e.g. the personal name and surname of an individual, the name of an organization) of the entity represented by this card. The purpose of this property is to define a name, even if the individual name components are not known. In addition, it is meant to provide alternative versions of the name for internationalisation. Implementations SHOULD prefer using the _name_ property over this one and SHOULD NOT store the concatenated name component values in this property.

2.2.3. nickNames

Type: String[] (optional).

The nick names of the entity represented by this card.
2.2.4. organizations

Type: Id[Organization] (optional).

The companies or organization names and units associated with this card. An Organization object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Organization.
* name: String (mandatory). The name of this organization.
* units: String[] (optional). Additional levels of organizational unit names.

2.2.5. titles

Type: Id[Title] (optional).

The job titles or functional positions of the entity represented by this card. A Title has object the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Title.
* title: String (mandatory). The title of the entity represented by this card.
* organization: Id (optional). The id of the organization in which this title is held.

2.2.6. speakToAs

Type: SpeakToAs (optional).

Provides information how to address, speak to or refer to the entity that is represented by this card. A SpeakToAs object has the following properties, of which at least one property other than @type MUST be set:

* @type: String (mandatory). Specifies the type of this object. This MUST be SpeakToAs.
* grammaticalGender: String (optional). Defines which grammatical gender to use in salutations and other grammatical constructs. Allowed values are:
  - animate
- female
- inanimate
- male
- neuter

Note that the grammatical gender does not allow to infer the gender identities or biological sex of the contact.

* pronouns: String (optional). Defines the gender pronouns that the contact chooses to use for themselves. Any value or form is allowed. Examples in English include she/her and they/them/their.

The property values SHOULD be localized in the language defined in the language property. They MAY be overridden in the localizations property (Section 2.5.1).

2.3. Contact and Resource properties

2.3.1. emails

Type: Id[EmailAddress] (optional).

The email addresses to contact the entity represented by this card. An EmailAddress object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be EmailAddress.

* email: String (mandatory). The email address. This MUST be an _addr-spec_ value as defined in Section 3.4.1 of [RFC5322].

* contexts: Context[Boolean] (optional) The contexts in which to use this email address. The value for each key in the object MUST be true.

* pref: Preference (optional) The preference of this email address in relation to other email addresses.

* label: String (optional). A label describing the value in more detail.
2.3.2. phones

Type: Id[Phone] (optional).

The phone numbers to contact the entity represented by this card. A Phone object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Phone.

* phone: String (mandatory). The phone value, as either a URI or a free-text phone number. Typical URI schemes are the [RFC3966] tel or [RFC3261] sip schemes, but any URI scheme is allowed.

* features: String[Boolean] (optional). The set of contact features that this phone number may be used for. The set is represented as an object, with each key being a method type. The value for each key in the object MUST be true. The method type MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:

  - voice The number is for calling by voice.
  - fax The number is for sending faxes.
  - pager The number is for a pager or beeper.
  - text The number supports text messages (SMS).
  - cell The number is for a cell phone.
  - textphone The number is for a device for people with hearing or speech difficulties.
  - video The number supports video conferencing.

* contexts: Context[Boolean] (optional) The contexts in which to use this number. The value for each key in the object MUST be true.

* pref: Preference (optional) The preference of this number in relation to other numbers.

* label: String (optional). A label describing the value in more detail.
2.3.3. online

Type: Id[Resource] (optional).

The online resources and services that are associated with the entity represented by this card. A Resource object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Resource.

* resource: String (mandatory). The resource value, where the allowed value form is defined by the the _type_ property. If the value form is URI, then the property value MUST be a valid _URI_ as defined in Section 3 of [RFC3986] and updates. In any case the value MUST NOT be empty.

* type: String (optional). The type of the resource. Allowed values are:

  - audio The resource is a digital sound URI, such as a recording of proper pronunciation of the name of this card.

  - calendar The resource is a calendar URI associated with the entity represented by this card.

  - contact The resource is an alternative contact method URI to contact the entity represented by this card. This typically is a web contact form using the "https" URI scheme, but may be any other.

  - directory The resource is a directory service URI where the entity represented by this card can be found in. This typically is an organizational directory that also contains associated entities, e.g. co-workers and management in a company directory.

  - directorySource The resource is a directory service entry URI of the entity represented by this card. In contrast to the "directory" type this resource only provides the means to access directory information for this entity.

  - freeBusy The resource is a free-busy calendaring URI for the entity represented by this card.

  - logo The resource is a graphic logo URI associated with the entity represented by this card.
- publicKey The resource is a cryptographic public key or certificate URI associated with the entity represented by this card.

- uri The resource value is any URI, e.g. a website link.

- username The resource value is a username associated with the entity represented by this card (e.g. for social media, or an IM client). The _label_ property SHOULD be included to identify what service this is for. For compatibility between clients, this label SHOULD be the canonical service name, including capitalisation. e.g. Twitter, Facebook, Skype, GitHub, XMPP. The resource value may be any non-empty free text.

* mediaType: String (optional). Used for URI resource values. Provides the media type [RFC2046] of the resource identified by the URI.

* contexts: Context[Boolean] (optional) The contexts in which to use this resource. The value for each key in the object MUST be true.

* pref: Preference (optional) The preference of this resource in relation to other resources.

* label: String (optional). A label describing the value in more detail.

2.3.4. scheduling

Type: Id[Scheduling] (optional).

The methods by which the entity receives calendar scheduling invitations and updates. A Scheduling object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Scheduling.

* sendTo: String[String] (mandatory). The keys in the property value are the available methods for scheduling. The value is a URI for the method specified in the key. This MUST be a valid "sendTo" property value as specified in Section 4.4.6 of [RFC8984] and updates.

* pref: Preference (optional) The preference of this scheduling object in relation to other scheduling objects.
2.3.5. photos

Type: Id[File] (optional).

A map of photo ids to File objects that contain photographs or images associated with this card. A typical use case is to include an avatar for display along the contact name.

A File object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be File.

* href: String (mandatory). A URI where to fetch the data of this file.

* mediaType: String (optional). The content-type of the file, if known.

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

* pref: Preference (optional) The preference of this photo in relation to other photos.

* label: String (optional). A label describing the value in more detail.

2.3.6. preferredContactMethod

Type : String (optional)

Defines the preferred method to contact the holder of this card. The value MUST be the property names: emails, phones, online.

2.3.7. preferredContactLanguages

Type : String[ContactLanguage[]] (optional)

Defines the preferred languages for contacting the entity associated with this card. The keys in the object MUST be [RFC5646] language tags. The values are a (possibly empty) list of contact language preferences for this language. A valid ContactLanguage object MUST have at least one of its properties set.

A ContactLanguage object has the following properties:
* @type: String (mandatory). Specifies the type of this object. This MUST be ContactLanguage.

* context: Context (optional). Defines the context in which to use this language.

* pref: Preference (optional). Defines the preference of this language in relation to other languages of the same context.

Also see the definition of the VCARD LANG property (Section 6.4.4., [RFC6350]).

2.4. Address and Location properties

2.4.1. addresses

Type: Id[Address] (optional).

A map of address ids to Address objects, containing physical locations. An Address object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Address.

* fullAddress: String (optional). The complete address, excluding type and label. This property is mainly useful to represent addresses of which the individual address components are unknown, or to provide localized representations.

* street: StreetComponent[] (optional). The street address. The concatenation of the component values, separated by whitespace, SHOULD result in a valid street address for the address locale. Doing so, implementations MAY ignore any separator components. The StreetComponent object type is defined in the paragraph below.

* locality: String (optional). The city, town, village, post town, or other locality within which the street address may be found.

* region: String (optional). The province, such as a state, county, or canton within which the locality may be found.

* country: String (optional). The country name.

* postcode: String (optional). The postal code, post code, ZIP code or other short code associated with the address by the relevant country’s postal system.

* countryCode: String (optional). The ISO-3166-1 country code.

* timeZone: String (optional) Identifies the time zone this address is located in. This either MUST be a time zone name registered in the IANA Time Zone Database (https://www.iana.org/time-zones), or it MUST be a valid TimeZoneId as defined in [RFC8984]. For the latter, a corresponding time zone MUST be defined in the timeZone property.

* contexts: Context[Boolean] (optional). The contexts of the address information. In addition to the common contexts, allowed values are:
  - billing An address to be used for billing.
  - postal An address to be used for delivering physical items. The value for each key in the object MUST be true.

* pref: Preference (optional) The preference of this address in relation to other addresses.

* label: String (optional). A label describing the value in more detail.

A StreetComponent object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be StreetComponent.

* type: String (mandatory). The type of this street component. The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:
  - name. The street name.
  - number. The street number.
  - apartment. The apartment number or identifier.
  - room. The room number or identifier.
  - extension. The extension designation or box number.
  - direction. The cardinal direction, e.g. "North".
  - building. The building or building part this address is located in.
- floor. The floor this address is located on.
- postOfficeBox. The post office box number or identifier.
- separator. A separator for two street components. The value property of the component includes the verbatim separator, for example a newline character.
- unknown. A name component value for which no type is known.

* value: String (mandatory). The value of this street component.

2.5. Multilingual properties

2.5.1. localizations

Type: String[PatchObject] (optional).

A map of language tags [RFC5646] to patches, which localize a property value into the locale of the respective language tag. The paths in the PatchObject keys are relative to the Card object that includes the localizations property. A patch MUST NOT target the localizations property.

The following example shows a Card object, where one of its addresses Tokyo is localized for the jp locale.

```json
"@type": "Card",
...
"addresses": {
  "addr1": {
    "@type": "Address",
    "locality": "Tokyo",
  }
},
"localizations": {
  "jp": {
    "addresses/addr1/locality": ""
  }
}
```

Figure 1
2.6. Additional properties

2.6.1. anniversaries

Type: Id[Anniversary] (optional).

These are memorable dates and events for the entity represented by this card. An Anniversary object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be Anniversary.

* type: String (optional). Specifies the type of the anniversary. This RFC predefines the following types, but implementations MAY use additional values:
  - birth: a birth day anniversary
  - death: a death day anniversary

* date: String (mandatory). The date of this anniversary, in the form "YYYY-MM-DD" (any part may be all 0s for unknown) or a [RFC3339] timestamp.

* place: Address (optional). An address associated with this anniversary, e.g. the place of birth or death.

* label: String (optional). A label describing the value in more detail.

2.6.2. personalInfo

Type: Id[PersonalInformation] (optional).

Defines personal information about the entity represented by this card. A PersonalInformation object has the following properties:

* @type: String (mandatory). Specifies the type of this object. This MUST be PersonalInformation.

* type: String (mandatory). Specifies the type for this personal information. Allowed values are:
  - expertise: a field of expertise or credential
  - hobby: a hobby
  - interest: an interest
2.6.3. notes

Type: String (optional).

Arbitrary notes about the entity represented by this card.

2.6.4. categories

Type: String[Boolean] (optional). The set of free-text or URI categories that relate to the card. The set is represented as an object, with each key being a category. The value for each key in the object MUST be true.

2.6.5. timeZones

Type: String[TimeZone] (optional). Maps identifiers of custom time zones to their time zone definitions. For a description of this property see the timeZones property definition in [RFC8984].

3. CardGroup

MIME type: application/jscontact+json;type=cardgroup

A CardGroup object represents a group of cards. Its members may be Cards or CardGroups.

3.1. Group properties

3.1.1. @type

Type: String (mandatory).

Specifies the type of this object. This MUST be CardGroup.
3.1.2. uid

Type: String (mandatory). The uid of this group. Both CardGroup and Card share the same namespace for the uid property.

3.1.3. members

Type: String[Boolean] (mandatory). The members of this group.

The set is represented as an object, with each key being the uid of another Card or CardGroup. The value for each key in the object MUST be true.

3.1.4. name

Type: String (optional). The user-visible name for the group, e.g. "Friends". This may be any UTF-8 string of at least 1 character in length and maximum 255 octets in size. The same name may be used by two different groups.

3.1.5. card

Type: Card (optional). The card that represents this group.

4. Implementation Status

NOTE: Please remove this section and the reference to [RFC7942] prior to publication as an RFC. This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist. According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

4.1. IIT-CNR/Registro.it
5. IANA Considerations

TBD

6. Security Considerations

TBD

7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Robert Stepanek
FastMail
PO Box 234, Collins St West
Melbourne VIC 8007
Australia
Email: rsto@fastmailteam.com

Mario Loffredo
IIT-CNR
Via Moruzzi, 1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it
JSContact: Converting from and to vCard
draft-ietf-calext-jscontact-vcard-01

Abstract

This document defines how to convert contact information as defined
in the JSContact [I-D.ietf-calext-jscontact] specification from and
to vCard.

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 13 October 2022.

Copyright Notice

Copyright (c) 2022 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
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 Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.
Table of Contents

1. Introduction .................................................. 3
   1.1. Motivation ................................................. 3
   1.2. Scope and Caveats .......................................... 4
   1.3. Conventions Used in This Document ......................... 4
   1.4. Extensions ................................................ 4

2. Translating vCard properties to JSContact .................... 5
   2.1. Common Parameters ......................................... 5
   2.2. Unmapped JSContact Information ............................ 5
   2.3. General Properties ......................................... 5
       2.3.1. BEGIN and END ....................................... 6
       2.3.2. SOURCE .............................................. 6
       2.3.3. KIND ................................................. 6
       2.3.4. XML ................................................ 7
   2.4. Identification Properties .................................. 7
       2.4.1. FN .................................................. 7
       2.4.2. N and NICKNAME ..................................... 7
       2.4.3. PHOTO ............................................... 9
       2.4.4. BDAY, BIRTHPLACE, DEATHDATE, DEATHPLACE,
             ANNIVERSARY ......................................... 9
       2.4.5. GENDER .............................................. 12
   2.5. Delivery Addressing Properties ............................ 13
       2.5.1. ADR .................................................. 13
   2.6. Communications Properties ................................ 15
       2.6.1. TEL ................................................ 16
       2.6.2. EMAIL .............................................. 16
       2.6.3. IMPP ............................................... 17
       2.6.4. LANG ............................................... 18
   2.7. Geographical Properties .................................. 19
       2.7.1. Time Zone Representation ............................. 20
   2.8. Organizational Properties ................................ 20
       2.8.1. TITLE and ROLE .................................... 20
       2.8.2. LOGO ............................................... 21
       2.8.3. ORG ............................................... 22
       2.8.4. MEMBER ............................................ 23
       2.8.5. RELATED ............................................ 25
       2.8.6. CONTACT-URI ....................................... 26
   2.9. Personal Information Properties .......................... 26
       2.9.1. EXPERTISE ......................................... 27
       2.9.2. HOBBY ............................................. 27
       2.9.3. INTEREST .......................................... 28
       2.9.4. ORG-DIRECTORY .................................... 29
   2.10. Explanatory Properties ................................. 30
       2.10.1. CATEGORIES ....................................... 30
       2.10.2. NOTE .............................................. 31
       2.10.3. PRODID ............................................ 32
       2.10.4. REV .............................................. 32
1. Introduction

1.1. Motivation

The JSContact specification [I-D.ietf-calext-jscontact] has been defined to represent contact card information as a more efficient alternative to vCard [RFC6350] and its JSON-based version named jCard [RFC7095].

While new applications might adopt JSContact as their main format to exchange contact card data, they are likely to interoperate with services and clients that just support vCard/jCard. Similarly, existing contact data providers and consumers already using vCard/jCard might want to represent their data also according to the JSContact specification.
To facilitate this, this document defines how to convert contact information as defined in the JSContact [I-D.ietf-calext-jscontact] specification from and to vCard.

1.2. Scope and Caveats

JSContact and vCard have a lot of semantics in common, however some differences must be outlined:

* The JSContact data model defines some contact information that doesn't have a direct mapping with vCard properties. In particular, unlike vCard, JSContact distinguishes between a single contact card, named Card, and a group of contact cards, named CardGroup.

* The properties that can be present multiple times in a vCard are represented through different collections in JSContact; mainly as maps, sometimes as lists, in some cases condensed in a single value.

1.3. Conventions Used in This Document

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].

In the following of this document, the vCard features, namely properties and parameters, are written in uppercase while the Card/CardGroup features are written in camel case wrapped in double quotes.

1.4. Extensions

While translating vCard to JSContact, any vCard property that doesn't have a direct counterpart in JSContact MUST be converted into a property whose name is prefixed by "ietf.org:<RFC defining the extension>:" (e.g. "ietf.org:rfc6350:").

Any custom extension MAY be added and its name MUST be prefixed with a specific domain name to avoid conflict, e.g. "example.com:customprop".

Likewise, while translating JSContact to vCard, a JSContact property that doesn't have a direct counterpart in vCard MUST be converted into a property whose name is prefixed with "X-" as specified in Section 6.10 of [RFC6350].
2. Translating vCard properties to JSContact

This section contains the translation rules from vCard to Card/CardGroup. The vCard properties are grouped according to the categories as defined in [RFC6350].

If a vCard represents a group of contacts, those vCard properties which don't have a counterpart in CardGroup are converted into related properties of the "CardGroup.card" object. In this case, the "uid" member of both the resulting CardGroup object and its "card" member MUST have the same value.

2.1. Common Parameters

The following mapping rules apply to parameters that are common to most of the vCard properties:

* The generic values of the TYPE parameter are mapped to the values of the "Context" type as defined in Section 1.5.1 of [I-D.ietf-calext-jscontact]. The "home" value corresponds to the "private" key. The mapping of those specific TYPE values used in the TEL and RELATED properties are defined in Section 2.6.1 and Section 2.8.5.
* The PREF parameter is mapped to the "pref" property.
* The MEDIATYPE parameter is mapped to the "mediaType" property. As described in Section 5.7 of [RFC6350], the media type of a resource can be identified by its URI. For example, "image/gif" can be derived from the ".gif" extension of a GIF image URI. JSContact producers MAY provide the media type information even when it is not specified in the vCard.
* The ALTID and LANGUAGE parameters are used in combination for associating the language-dependent alternatives with a given property. Such alternatives are represented by using the "localizations" map: the "localizations" key is the LANGUAGE value, the key of the related PatchObject map is the JSON pointer of the JSContact member matching the vCard property while the value is the JSContact member itself.

2.2. Unmapped JSContact Information

The rules to generate a map key of type Id as well as a value for "created", "language" and "preferredContactMethod" properties are out of the scope of this document.

2.3. General Properties
2.3.1. BEGIN and END

The BEGIN and END properties don’t have a direct match with a JSContact feature.

2.3.2. SOURCE

A SOURCE property is represented as an entry of the "online" map (Figure 1). The entry value is a "Resource" object whose "type" member is set to "directorySource" and the "resource" member is the SOURCE value.

The PREF and MEDIATYPE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0
...
SOURCE:http://directory.example.com/addressbooks/jdoe/Jean%20Dupont.vcf
...
END:VCARD

{
  "@type": "Card",
  ...
  "online":{
    ...
    "a-source":{
      "@type": "Resource",
      "type": "directorySource",
      "resource": "http://directory.example.com/addressbooks/jdoe/Jean%20Dupont.vcf"
    },
    ...
  },
  ...
}

Figure 1: SOURCE mapping example

2.3.3. KIND

The KIND property is mapped to the "kind" member (Figure 2). Allowed values are those described in Section 6.1.4 of [RFC6350] and extended with the values declared in [RFC6473] and [RFC6869]. The value "group" is reserved for a CardGroup instance.
BEGIN:VCARD
VERSION:4.0
...
KIND:individual
...
END:VCARD

{
  "@type": "Card",
  ...
  "kind": "individual",
  ...
}

Figure 2: KIND mapping example

2.3.4. XML

The XML property doesn’t have a direct match with a JSContact feature.

2.4. Identification Properties

2.4.1. FN

All the FN instances are represented through the "fullName" member (Figure 3). The presence of multiple instances is implicitly associated with the full name translations in various languages regardless of the presence of the ALTID parameter. Each translation is mapped according to the rules as defined in Section 2.1.

If the vCard represents a group of contacts, implementers MAY convert the FN property into either "CardGroup.card.fullName" or "CardGroup.name" or both properties.

2.4.2. N and NICKNAME

The N instances are converted into equivalent items of the "components" array of the "name" property (Figure 3): the N components are transformed into related "NameComponent" objects as presented in Table 1. Name components SHOULD be ordered such that their values joined by whitespace produce a valid full name of this entity.

Each NICKNAME instance is mapped to an item of "nickNames" array.
Table 1: N components mapping

<table>
<thead>
<tr>
<th>N component</th>
<th>&quot;type&quot; value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honorific Prefixes</td>
<td>prefix</td>
</tr>
<tr>
<td>Given Names</td>
<td>personal</td>
</tr>
<tr>
<td>Family Names</td>
<td>surname</td>
</tr>
<tr>
<td>Additional Names</td>
<td>additional</td>
</tr>
<tr>
<td>Honorific Suffixes</td>
<td>suffix</td>
</tr>
</tbody>
</table>

BEGIN:VCARD
VERSION:4.0
...
FN:Mr. John Q. Public, Esq.
N:Public;John;Quinlan;Mr.;Esq.
NICKNAME:Johnny
...
END:VCARD

{
  "@type": "Card",
  ...
  "fullName": "Mr. John Q. Public, Esq.",
  "name":{
    "@type": "Name",
    "components":[
      { "@type": "NameComponent", "type": "prefix", "value":"Mr." },
      { "@type": "NameComponent", "type": "personal", "value":"John" },
      { "@type": "NameComponent", "type": "surname", "value":"Public" },
      { "@type": "NameComponent", "type": "additional", "value":"Quinlan" },
      { "@type": "NameComponent", "type": "suffix", "value":"Esq." }
    ]
  },
  "nickNames":[
    "Johnny"
  ],
  ...
}

Figure 3: FN, N, NICKNAME mapping example
2.4.3. PHOTO

A PHOTO property is represented as an entry of the "photos" map (Figure 4). The entry value is a "File" object whose "href" member is the PHOTO value.

The PREF and MEDIATYPE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0
...
PHOTO:http://www.example.com/pub/photos/jqpublic.gif
...
END:VCARD

{
"@type": "Card",
...
"photos":{
...
"a-photo":{
"@type": "File",
"href": "http://www.example.com/pub/photos/jqpublic.gif"
},
...
},
...
}

Figure 4: PHOTO mapping example

2.4.4. BDAY, BIRTHPLACE, DEATHDATE, DEATHPLACE, ANNIVERSARY

The BDAY and ANNIVERSARY properties and the extensions BIRTHPLACE, DEATHDATE, DEATHPLACE described in [RFC6350] are represented as "Anniversary" objects included in the "anniversaries" map (Figure 5):

* BDAY and BIRTHPLACE are mapped to "date" and "place" where "type" is set to "birth";
* DEATHDATE and DEATHPLACE are mapped to "date" and "place" where "type" is set to "death";
* ANNIVERSARY is mapped to "date" where "type" is empty and "label" is set to a value describing in detail the kind of anniversary (e.g. "marriage date" for the wedding anniversary).

Both birth and death places are represented as instances of the "Address" object.
The BIRTHPLACE and DEATHPLACE properties that are represented as geo
URIs are converted into "Address" instances including only the
"coordinates" member. If the URI value is not a geo URI, the place
is ignored.

The ALTID and LANGUAGE parameters of both BIRTHPLACE and DEATHPLACE
properties are mapped according to the rules as defined in
Section 2.1.
BEGIN:VCARD
VERSION:4.0
... 
BDAY:19531015T231000Z
BIRTHPLACE:Mail Drop: TNE QB\n123 Main Street\nAny Town, CA 91921-1234\nU.S.A.
DEATHDATE:19960415
DEATHPLACE:4445 Courtright Street\nNew England, ND 58647\nU.S.A.
ANNIVERSARY:19860201
...
END:VCARD

{
  "@type": "Card",
  ...
  "anniversaries": {
    "ANNIVERSARY-1" : {
      "@type": "Anniversary",
      "type": "birth",
      "date": "1953-10-15T23:10:00Z",
      "place":{
        "@type": "Address",
        "fullAddress": "Mail Drop: TNE QB\n123 Main Street\nAny Town, CA 91921-1234\nU.S.A."
      }
    },
    "ANNIVERSARY-2" : {
      "@type": "Anniversary",
      "type": "death",
      "date": "1996-04-15",
      "place":{
        "@type": "Address",
        "fullAddress": "4445 Courtright Street\nNew England, ND 58647\nU.S.A."
      }
    },
    "ANNIVERSARY-3" : {
      "@type": "Anniversary",
      "label": "marriage date",
      "date": "1986-02-01"
    }
  }
}
...

Figure 5: BDAY, BIRTHPLACE, DEATHDATE, DEATHPLACE, ANNIVERSARY
mapping example
2.4.5. GENDER

The GENDER property is a single structured value with two optional components: the biological sex and the gender information. The former is represented as an enumerated value, while the latter as a free-form text. As opposed to such a representation, the JSContact specification includes the "SpeakToAs" object just to represent how to address, speak to or refer to the contact. In particular, some pre-defined values are allowed for the "grammaticalGender" member.

For the reasons stated above, the GENDER property doesn’t have a direct match with the "SpeakToAs" object. However, on the assumption that the GENDER property doesn’t store the actual biological sex of the contact, implementations MAY use the conversion rules shown in Table 2 and Table 3.

<table>
<thead>
<tr>
<th>GENDER value</th>
<th>&quot;SpeakToAs.grammaticalGender&quot; value</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>male</td>
</tr>
<tr>
<td>F</td>
<td>female</td>
</tr>
<tr>
<td>N</td>
<td>neuter</td>
</tr>
<tr>
<td>O</td>
<td>animate</td>
</tr>
<tr>
<td>U</td>
<td>SpeakToAs = null</td>
</tr>
</tbody>
</table>

Table 2: GENDER to SpeakToAs conversion

<table>
<thead>
<tr>
<th>&quot;SpeakToAs.grammaticalGender&quot; value</th>
<th>GENDER value</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>M</td>
</tr>
<tr>
<td>female</td>
<td>F</td>
</tr>
<tr>
<td>neuter</td>
<td>N</td>
</tr>
<tr>
<td>animate</td>
<td>O</td>
</tr>
<tr>
<td>inanimate</td>
<td>N;inanimate</td>
</tr>
</tbody>
</table>

Table 3: SpeakToAs to GENDER conversion
2.5. Delivery Addressing Properties

2.5.1. ADR

An ADR property is represented as an entry of the "addresses" map (Figure 6). The entry value is an "Address" object.

The ADR components are transformed into the "Address" members as presented in Table 4 and Table 5.

The "street address" and "extended address" ADR components MAY be converted into either a single StreetComponent item or a combination of StreetComponent items.

<table>
<thead>
<tr>
<th>ADR component</th>
<th>Address member</th>
</tr>
</thead>
<tbody>
<tr>
<td>locality</td>
<td>locality</td>
</tr>
<tr>
<td>region</td>
<td>region</td>
</tr>
<tr>
<td>postal code</td>
<td>postcode</td>
</tr>
<tr>
<td>country name</td>
<td>country</td>
</tr>
</tbody>
</table>

Table 4: ADR components vs. Address members mapping

<table>
<thead>
<tr>
<th>ADR component</th>
<th>Single StreetComponent item</th>
<th>Combination of StreetComponent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>post office box</td>
<td>postOfficeBox</td>
<td></td>
</tr>
<tr>
<td>extended address</td>
<td>extension</td>
<td>extension, building, floor, room, apartment</td>
</tr>
<tr>
<td>street address</td>
<td>name</td>
<td>name, number, direction</td>
</tr>
</tbody>
</table>

Table 5: ADR components vs. StreetComponent items mapping

The LABEL parameter is converted into the "fullAddress" member.

The GEO parameter is converted into the "coordinates" member.
The TZ parameter is converted into the "timeZone" member.

The CC parameter as defined in [RFC8605] is converted into the "countryCode" member.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

The ALTID and LANGUAGE parameters are mapped according to the rules as defined in Section 2.1. Each possible language-dependent alternative is represented as an entry of the PatchObject map where the key references the "fullAddress" member.
BEGIN:VCARD
VERSION:4.0

ADR;TYPE=work;CC=US;;54321 Oak St;Reston;VA;20190;USA
ADR;TYPE=home;CC=US;;12345 Elm St;Reston;VA;20190;USA

END:VCARD

{
  "@type": "Card",
  ...
  "addresses":{
    "work-address":{
      "@type": "Address",
      "contexts":{ "work": true },
      "fullAddress": "54321 Oak St\nReston\nVA\n20190\nUSA",
      "street": [
        { "@type": "StreetComponent", "type": "name", "value": "Oak St" },
        { "@type": "StreetComponent", "type": "number", "value": "54321" }
      ],
      "locality": "Reston",
      "region": "VA",
      "country": "USA",
      "postcode": "20190",
      "countryCode": "US"
    },
    "private-address":{
      "@type": "Address",
      "contexts":{ "private": true },
      "fullAddress": "12345 Elm St\nReston\nVA\n20190\nUSA",
      "street": [
        { "@type": "StreetComponent", "type": "name", "value": "Elm St" },
        { "@type": "StreetComponent", "type": "number", "value": "12345" }
      ],
      "locality": "Reston",
      "region": "VA",
      "country": "USA",
      "postcode": "20190",
      "countryCode": "US"
    }
  }
}

Figure 6: ADR mapping example

2.6. Communications Properties
2.6.1. TEL

A TEL property is represented as an entry of the "phones" map (Figure 7). The entry value is a "Phone" object. The TEL-specific values of the TYPE parameter are mapped to the "features" map keys. The values that don't match a key are represented as comma-separated values of the "label" member. The "phone" member is set to the TEL value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0

... TEL;VALUE=uri;PREF=1;TYPE="voice,home":tel:+1-555-555-5555;ext=5555
TEL;VALUE=uri;TYPE=home:tel:+33-01-23-45-67
...
END:VCARD

{ "@type": "Card", "phones": { "a-phone": { "@type": "Phone", "contexts": { "private": true }, "features": { "voice": true }, "phone": "tel:+1-555-555-5555;ext=5555", "pref": 1 }, "another-phone": { "@type": "Phone", "contexts": { "private": true }, "phone": "tel:+33-01-23-45-67" } }, ...
}

Figure 7: TEL mapping example

2.6.2. EMAIL

An EMAIL property is represented as an entry of the "emails" map (Figure 8). The entry value is an "EmailAddress" object. The "email" member is set to the EMAIL value.
The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

```
BEGIN:VCARD
VERSION:4.0
...
EMAIL;TYPE=work:jqpublic@xyz.example.com
EMAIL;PREF=1:jane_doe@example.com
...
END:VCARD

{
  "@type": "Card",
  ...
  "emails":{
    "work-email":{
      "@type": "EmailAddress",
      "contexts":{ "work": true },
      "email": "jqpublic@xyz.example.com"
    },
    "private-email":{
      "@type": "EmailAddress",
      "email": "jane_doe@example.com",
      "pref": 1
    }
  },
  ...
}
```

Figure 8: EMAIL mapping example

### 2.6.3. IMPP

An IMPP property is represented as an entry of the "online" map (Figure 9). The entry value is a "Resource" object whose "type" member is set to "username", the "label" member is set to "XMPP" and the "resource" member is the IMPP value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0
...
IMPP;PREF=1:xmpp:alice@example.com
...
END:VCARD

{
"@type": "Card",
...
"online":{
...
{  
"@type": "Resource",
"type": "username",
"label": "XMPP",
"value": "alice@example.com",
"pref": 1
},
...
},
...
...

Figure 9: IMPP mapping example

2.6.4. LANG

A LANG property is represented as an entry of the "preferredContactLanguages" map (Figure 10). The entry keys correspond to the language tags, the corresponding entry values are arrays of "ContactLanguage" objects.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

If both PREF and TYPE parameters are missing, the array of "ContactLanguage" objects MUST be empty.
BEGIN:VCARD
VERSION:4.0
...
LANG;TYPE=work;PREF=1:en
LANG;TYPE=work;PREF=2:fr
LANG;TYPE=home:fr
...
END:VCARD

{
  "@type": "Card",
  ...
  "preferredContactLanguages":{
    "en":[
    {
      "@type": "ContactLanguage",
      "context": "work",
      "pref": 1
    }
    ],
    "fr":[
    {
      "@type": "ContactLanguage",
      "context": "work",
      "pref": 2
    },
    {
      "@type": "ContactLanguage",
      "context": "private"
    }
    ],
  },
  ...
}

Figure 10: LANG mapping example

2.7. Geographical Properties

The GEO and TZ properties are not directly mapped to topmost Card members because the same information is represented through equivalent "Address" members.

The ALTID parameter is used for associating both GEO and TZ properties with the related address information. When the ALTID parameter is missing, the matched members SHOULD be included in the first "Address" object.
2.7.1.  Time Zone Representation

As specified in Section 6.5.1 of [RFC6350], the time zone information can be represented as a time zone name, as a UTC offset or as a URI.

* If the TZ value is defined in the IANA timezone database, it is directly matched by the "timeZone" member in JSContact.
* An UTC offset MUST be converted into the related "Etc/GMT" time zone (e.g. the value "-0500" converts to "Etc/GMT+5"). If the UTC offset value contains minutes information or is not an IANA timezone name, it requires special handling.
* Since there is no URI scheme defined for time zones [uri-schemes], any implementation that does use some a custom URI for a time zone is not interoperable anyway. In this case, if the URI corresponds to an IANA time zone [time-zones], this latter SHOULD be used. Otherwise, the URI value is dumped into a string.

2.8.  Organizational Properties

2.8.1.  TITLE and ROLE

Both TITLE and ROLE properties are represented as entries of the "titles" map (Figure 11). The entry value is a "Title" object whose "title" member includes information about the title or role. The rules to set the "organization" member are out of the scope of this document.

The ALTID and LANGUAGE parameters are mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0
...
TITLE:Research Scientist
ROLE:Project Leader
...
END:VCARD

{  
  "@type": "Card",
  ...
  "titles":{
    "a-title":{
      "@type": "Title",
      "title": "Project Leader"
    },
    "another-title":{
      "@type": "Title",
      "title": "Research Scientist"
    }
  }
}

Figure 11: TITLE and ROLE mapping example

2.8.2. LOGO

A LOGO property is represented as an entry of the "online" map (Figure 12). The entry value is a "Resource" object whose "type" member is set to "logo" and the "resource" member is the LOGO value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0
...
LOGO:http://www.example.com/pub/logos/abccorp.jpg
...
END:VCARD

{
  "@type": "Card",
  ...
  "online":{
    ...
    "a-logo":{
      "@type": "Resource",
      "type": "logo",
      "resource": "http://www.example.com/pub/logos/abccorp.jpg"
    },
    ...
  },
  ...
}

Figure 12: LOGO mapping example

2.8.3. ORG

An ORG property is represented as an entry of the "organizations" map (Figure 13). The entry value is an "Organization" object whose "name" member contains the organizational name and the "units" member contains the organizational units.

The ALTID and LANGUAGE parameters are mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0
...
ORG:ABC, Inc.;North American Division;Marketing
...
END:VCARD
{
  "@type": "Card",
  ...
  "organizations":{
    "an-organization":{
      "@type": "Organization",
      "name": "ABC, Inc.",
      "units":[
        "North American Division",
        "Marketing"
      ]
    }
  }
},
...
}

Figure 13: ORG mapping example

2.8.4. MEMBER

According to the JSContact specification, a group of contact cards is represented through a CardGroup (Figure 14). The uids of the contact cards composing the group are included in the "members" map.

In this case, the PREF parameter has not a JSContact counterpart; however, the implementers MAY insert the map entries by order of preference.
BEGIN:VCARD
VERSION:4.0
KIND:group
FN:The Doe family
MEMBER:urn:uuid:03a0e51f-d1aa-4385-8a53-e29025acd8af
MEMBER:urn:uuid:b8767877-b4a1-4c70-9acc-505d3819e519
END:VCARD

Figure 14: Group example

Only if the GROUP contains properties that don’t have a mapping to CardGroup properties, then the CardGroup.card property MAY contain the optional Card object of this group.

Figure 15: card member of CardGroup object
2.8.5. RELATED

All the RELATED instances are converted into the "relatedTo" map (Figure 16): an entry for each entity the entity described by the Card is associated with. The map keys are the "uid" values of the associated cards.

Each map value is a "Relation" object including only the "relation" member represented as a set of the RELATED-specific values of the TYPE parameter as defined in Section 6.6.6 of [RFC6350].

If the relation type is unspecified, the "relation" member MUST be empty.

BEGIN:VCARD
VERSION:4.0
...
RELATED;TYPE=friend:urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
RELATED;TYPE=contact:http://example.com/directory/jdoe.vcf
RELATED;VALUE=text:Please contact my assistant Jane Doe for any inquiries.
...
END:VCARD

{
 "@type": "Card",
...
 "relatedTo":{
  {
   "@type": "Relation",
   "urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6":{
    "relation":{ "friend": true }
   }
  },
  {
   "@type": "Relation",
   "http://example.com/directory/jdoe.vcf":{
    "relation":{ "contact": true }
   }
  },
  {
   "@type": "Relation",
   "Please contact my assistant Jane Doe for any inquiries.":{
    "relation":{ }
   }
  }
  },
...
}
2.8.6. CONTACT-URI

A CONTACT-URI property as defined in [RFC8605] is represented as an entry of the "online" map (Figure 17). The entry value is a "Resource" object whose "type" member is set to "contact" and the "resource" member is the CONTACT-URI value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

```
BEGIN:VCARD
VERSION:4.0
... CONTACT-URI;PREF=1:mailto:contact@example.com ...
END:VCARD

{
  "@type": "Card",
  ...
  "online": {
    ...
    "a-contact-uri": {
      "@type": "Resource",
      "type": "contact",
      "resource": "mailto:contact@example.com",
      "pref": 1
    },
    ...
  },
  ...
}
```

2.9. Personal Information Properties

The LEVEL parameter as defined in [RFC6715] is directly mapped to the "level" property of the "PersonalInformation" type apart from when LEVEL is used in the EXPERTISE property; in this case, the values are converted as in the following:

* "beginner" is converted into "low";
* "average" is converted into "medium";
* "expert" is converted into "high".
2.9.1. EXPERTISE

An EXPERTISE property as defined in [RFC6715] is represented as a "PersonalInformation" object in the "personalInfo" map (Figure 18). The "type" member is set to "expertise".

The INDEX parameter is represented as the index of the expertise among the declared expertises.

BEGIN:VCARD
VERSION:4.0
...
EXPERTISE;LEVEL=beginner;INDEX=2:chinese literature
EXPERTISE;INDEX=1;LEVEL=expert:chemistry
...
END:VCARD

{
  "@type": "Card",
  ...
  "personalInfo": {
    ...
    "PERSINFO-1": {
      "@type": "PersonalInformation",
      "type": "expertise",
      "value": "chemistry",
      "level": "high"
    },
    "PERSINFO-2": {
      "@type": "PersonalInformation",
      "type": "expertise",
      "value": "chinese literature",
      "level": "low"
    },
    ...
  }
}

Figure 18: EXPERTISE mapping example

2.9.2. HOBBY

A HOBBY property as defined in [RFC6715] is represented as a "PersonalInformation" object in the "personalInfo" map (Figure 19). The "type" member is set to "hobby".

{...
  ...
}
The INDEX parameter is represented as the index of the hobby among the declared hobbies.

BEGIN:VCARD
VERSION:4.0
...
HOBBY;INDEX=1;LEVEL=high:reading
HOBBY;INDEX=2;LEVEL=high:sewing
...
END:VCARD

{
"@type": "Card",
...
"personalInfo":{
...
"PERSINFO-1" : {
  "@type": "PersonalInformation",
  "type": "hobby",
  "value": "reading",
  "level": "high"
},
"PERSINFO-2" : {
  "@type": "PersonalInformation",
  "type": "hobby",
  "value": "sewing",
  "level": "high"
},
...
...}

Figure 19: HOBBY mapping example

2.9.3. INTEREST

An INTEREST property as defined in [RFC6715] is represented as a "PersonalInformation" object in the "personalInfo" map (Figure 20). The "type" member is set to "interest".

The INDEX parameter is represented as the index of the interest among the declared interests.
BEGIN:VCARD
VERSION:4.0
...
INTEREST;INDEX=1;LEVEL=medium:r&b music
INTEREST;INDEX=2;LEVEL=high:rock n roll music
...
END:VCARD

{
   "@type": "Card",
   ...
   "personalInfo":{
      ...
      "PERSINFO-1" : {
         "@type": "PersonalInformation",
         "type": "interest",
         "value": "r&b music",
         "level": "medium"
      },
      "PERSINFO-2" : {
         "@type": "PersonalInformation",
         "type": "interest",
         "value": "rock n roll music",
         "level": "high"
      },
      ...
   },
   ...
   ...
}

Figure 20: INTEREST mapping example

2.9.4.  ORG-DIRECTORY

An ORG-DIRECTORY property as defined in [RFC6715] is represented as an entry of the "online" map (Figure 21). The entry value is a "Resource" object whose "type" member is set to "directory" and the "resource" member is the ORG-DIRECTORY value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

The INDEX parameter is represented as the index of the directory among the online resources with the "directory" type.
BEGIN:VCARD
VERSION:4.0
...
ORG-DIRECTORY;INDEX=1:http://directory.mycompany.example.com
ORG-DIRECTORY;PREF=1:ldap://ldap.tech.example/o=Example%20Tech,ou=Engineering
...
END:VCARD
{
...
"online":{
   "@type": "Card",
   ...
   "an-org-directory":{
      "@type": "Resource",
      "type": "directory",
      "resource": "http://directory.mycompany.example.com"
   },
   "another-org-directory":{
      "@type": "Resource",
      "type": "directory",
      "resource": "ldap://ldap.tech.example/o=Example%20Tech,ou=Engineering",
      "pref": 1
   },
   ...
   ...
}

Figure 21: ORG-DIRECTORY mapping example

2.10. Explanatory Properties

2.10.1. CATEGORIES

A CATEGORIES property is converted into a set of entries of the "categories" map (Figure 22). The keys are the comma-separated text values of the CATEGORIES property.

In this case, the PREF parameter has not a JSContact counterpart; however, implementers MAY use a map preserving the order of insertion and the map entries can be inserted by order of preference.
BEGIN:VCARD
VERSION:4.0
...
CATEGORIES:INTERNET,IETF,INDUSTRY,INFORMATION TECHNOLOGY
...
END:VCARD
{
  "@type": "Card",
  ... ...
  "categories": { 
    "INTERNET": true,
    "IETF": true,
    "INDUSTRY": true,
    "INFORMATION TECHNOLOGY": true
  }, ...
}

Figure 22: CATEGORIES mapping example

2.10.2. NOTE

A NOTE property is mapped to the "notes" property (Figure 23). All the NOTE instances are condensed into a single note and separated by newline.

The ALTID and LANGUAGE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0
...
NOTE:This fax number is operational 0800 to 1715 EST, Mon-Fri.
...
END:VCARD
{
  "@type": "Card",
  ...
  "notes": "This fax number is operational 0800 to 1715 EST, Mon-Fri.",
  ... }

Figure 23: NOTE mapping example
2.10.3. PRODID

The PRODID property is converted into the "prodId" member (Figure 24).

```
BEGIN:VCARD
VERSION:4.0
...
PRODID:-//ONLINE DIRECTORY//NONSGML Version 1//EN
...
END:VCARD
{
 "@type": "Card",
...
"prodId": "-//ONLINE DIRECTORY//NONSGML Version 1//EN",
...
}
```

Figure 24: PRODID mapping example

2.10.4. REV

The REV property is transformed into the "updated" member (Figure 25).

```
BEGIN:VCARD
VERSION:4.0
...
REV:19951031T222710Z
...
END:VCARD
{
 "@type": "Card",
...
"updated": "1995-10-31T22:27:10Z",
...
}
```

Figure 25: REV mapping example

2.10.5. SOUND

A SOUND property is represented as an entry of the "online" map (Figure 26). The entry value is a "Resource" object whose "type" member is set to "audio" and the "resource" member is the SOUND value.
The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0
...
SOUND:CID:JOHNQPUBLIC.part8.19960229T080000.xyzMail@example.com
...
END:VCARD

{
"@type": "Card",
...
"online":{
...
"a-sound":{
"@type": "Resource",
"type": "audio",
"resource": "CID:JOHNQPUBLIC.part8.19960229T080000.xyzMail@example.com"
},
...
},
...
}

Figure 26: SOUND mapping example

2.10.6. UID

The UID property corresponds to the "uid" property (Figure 27) in both Card and CardGroup.

BEGIN:VCARD
VERSION:4.0
...
UID:urn:uuid:f81d4fae-7dec-11d0-00a0c91e6bf6
...
END:VCARD

{
"@type": "Card",
...
"uid": "urn:uuid:f81d4fae-7dec-11d0-00a0c91e6bf6",
...
}

Figure 27: UID mapping example
2.10.7. CLIENTPIDMAP and PDI Parameter

The CLIENTPIDMAP property and the PDI parameter don’t have a direct match with a Card feature.

2.10.8. URL

An URL property is represented as an entry of the "online" map (Figure 28). The entry value is a "Resource" object whose "type" member is set to "uri" and the "resource" member is the URL value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0
... URL:http://example.org/restaurant.french/~chezchic.html ...
END:VCARD

{
  "@type": "Card",
  ...
  "online":{
    ...
    "an-url":{
      "@type": "Resource",
      "type": "uri",
      "resource": "http://example.org/restaurant.french/~chezchic.html"
    },
    ...
  },
  ...
}

Figure 28: URL mapping example

2.10.9. VERSION

The VERSION property doesn’t have a direct match with a JSContact feature.

2.11. Security Properties
2.11.1.  KEY

A KEY property is represented as an entry of the "online" map (Figure 29). The entry value is a "Resource" object whose "type" member is set to "publicKey" and the "resource" member is the KEY value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.

BEGIN:VCARD
VERSION:4.0
...
KEY:http://www.example.com/keys/jdoe.cer
...
END:VCARD

{
  "@type": "Card",
  ...
  "online":{
    ...
    "a-key":{
      "@type": "Resource",
      "type": "publicKey",
      "resource": "http://www.example.com/keys/jdoe.cer"
    },
    ...
  },
  ...
}

Figure 29: KEY mapping example

2.12.  Calendar Properties

2.12.1.  FBURL

An FBURL property is represented as an entry of the "online" map (Figure 30). The entry value is a "Resource" object whose "type" member is set to "freeBusy" and the "resource" member is the FBURL value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0

FBURL;PREF=1:http://www.example.com/busy/janedoe
FBURL;MEDIATYPE=text/calendar:ftp://example.com/busy/project-a.ifb

END:VCARD

{
  "@type": "Card",
  ...
  "online": {
    ...
    "an-fburl": {
      "@type": "Resource",
      "type": "freeBusy",
      "resource": "http://www.example.com/busy/janedoe",
      "pref": 1
    },
    "another-fburl": {
      "@type": "Resource",
      "type": "freeBusy",
      "resource": "ftp://example.com/busy/project-a.ifb",
      "mediaType": "text/calendar"
    },
    ...
  },
  ...
}

Figure 30: FBURL mapping example

2.12.2. CALADRURI

A CALADRURI property is represented as an entry of the "scheduling" map (Figure 31). The entry value is a "Scheduling" object whose "sendTo" map includes an entry whose key is set to "imip" and value is set to the CALADRURI value.

The PREF parameter is mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0
... 
CALADRURI;PREF=1:mailto:janedoe@example.com
CALADRURI:http://example.com/calendar/jdoe
...
END:VCARD

{
"@type": "Card",
...
"scheduling":{
...
"a-caladruri":{
  "@type": "Scheduling",
  "sendTo": {
    "imip": "mailto:janedoe@example.com"
  },
  "pref": 1
 },
"another-caladruri":{
  "@type": "Scheduling",
  "sendTo": {
    "imip": "http://example.com/calendar/jdoe"
  },
},
...
},
...
}

Figure 31: CALADRURI mapping example

2.12.3.  CALURI

A CALURI property is represented as an entry of the "online" map (Figure 32). The entry value is a "Resource" object whose "type" member is set to "calendar" and the "resource" member is the CALURI value.

The PREF and TYPE parameters are mapped according to the rules as defined in Section 2.1.
BEGIN:VCARD
VERSION:4.0
...
CALURI;PREF=1:http://cal.example.com/calA
CALURI;MEDIATYPE=text/calendar:ftp://ftp.example.com/calA.ics
...
END:VCARD
{
"@type": "Card",
...
"online":{
...
"a-caluri":{
   "@type": "Resource",
   "type": "calendar",
   "resource": "http://cal.example.com/calA",
   "pref": 1
 },
"another-caluri":{
   "@type": "Resource",
   "type": "calendar",
   "resource": "ftp://ftp.example.com/calA.ics",
   "mediaType": "text/calendar"
 },
...
},
...
...

Figure 32: CALURI mapping example

2.13. vCard Unmatched Properties

The unmatched vCard properties MAY be converted into JSContact properties whose name contains the prefix "ietf.org:rfc6350:" followed by property name in uppercase (i.e. ietf.org:rfc6350:CLIENTPIDMAP”).


While converting a vCard into a Card/CardGroup, only the topmost "uid" member is mandatory. Implementers are REQUIRED to set it when it is missing.
3. Translating JSContact properties to vCard

In most of the cases, the rules about the translation from Card/CardGroup to vCard can be derived by reversing the rules presented in Section 2. The remaining cases are treated in the following of this section.

3.1. Id

Where a map key is of type Id, implementers are free to either ignore it or preserve it as a vCard information (e.g. a vCard parameter).

3.2. Localizations

Each PatchObject entry value of each "localizations" entry is converted into a instance of the vCard property matching the JSContact member referenced by the PatchObject entry key. The LANGUAGE parameter of such alternative MUST be set to the value of the given "localizations" entry. The LANGUAGE parameter of a vCard property presenting, at least, a language-dependent alternative MUST be set to the value of the JSContact "language" property if it is valued. Implementers MAY set the ALTID parameter to group language-based alternatives of the same value.

Note also that the components of some vCard values and their language-dependent alternatives are split into different JSContact values. For example, the "name" and "units" values for a given language must be grouped to make a single ORG value where components are separated by ";".

3.3. Date and Time Representations

The JSContact spec defines the "UTCDatetime" type to represent [RFC3339] "date-time" format with further restrictions. This means that the matched vCard format for a "UTCDatetime" value MUST be one of the formats shown in Section 4.3.5 of [RFC6350] (i.e. "19961022T140000Z").

In addition to such format, the "date" member of the "Anniversary" type allows to specify both dates without the time and partial dates. In such cases, the corresponding vCard format is that defined in Section 4.3.1.

3.4. Time Zone

The time zone name as represented by the "timeZone" property is mapped to the TZ parameter.
Implementers MAY map an "Etc/GMT" time zone either preserving the
time zone name or converting it into a UTC offset.

3.5. JSContact Types Matching Multiple vCard Properties

3.5.1. Title

The "titles" property contains information about the job, the
position or the role of the entity the card represents. In vCard,
such information is split into the TITLE and the ROLE properties.
This specification defines TITLE as the default target property when
converting the "titles" property.

3.5.2. Resource

The "online" property includes resources that are usually represented
through different vCard properties. The matched vCard property of a
"Resource" object can be derived from the value of its "type" member.

Any resource included in the "online" map that doesn’t match a vCard
property MAY be converted into a vCard extended property.

3.6. CardGroup

A CardGroup object is converted into a vCard by merging its
properties with the properties of "CardGroup.card" object. If the
"CardGroup.card.fullName" property exists, it MUST be used to set the
FN value.

3.7. Card Unmatched Properties

Both the "preferredContactMethod" and "created" members don’t match
any vCard property. Implementers MAY represent them as vCard
extended properties.

3.8. vCard Required Properties

While converting a Card/CardGroup into a vCard, only the FN property
is required. Since both the "Card.fullName" and "CardGroup.name"
properties are optional, implementers are REQUIRED to generate an FN
value when it is missing.

4. IANA Considerations

This document has no actions for IANA.
5. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol as defined in this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. CNR

* Responsible Organization: National Research Council (CNR) of Italy
* Location: https://github.com/consiglionazionaledellericerche/jscontact-tools
* Description: This implementation includes tools for JSContact creation, validation, serialization/deserialization, and conversion from vCard, xCard and jCard.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

6. Security Considerations

This document doesn’t present any security consideration.

7. References

7.1. Normative References
7.2. Informative References


7.2. Informative References
[I-D.ietf-calext-jscontact]

[time-zones]

[uri-schemes]

Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it
URI: http://www.iit.cnr.it

Robert Stepanek
FastMail
PO Box 234, Collins St West
Melbourne VIC 8007
Australia
Email: rsto@fastmailteam.com
URI: https://www.fastmail.com
Abstract

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

This specification also adds values to the Preferences Registry defined in RFC7240 to add the subscribe-enhanced-get and limit preferences and to the link relations directory defined in RFC8288.

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 22 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
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 lightweight 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 [RFC8288]. 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.
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

>> 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 3

This is an example of the subsequent request and response for an old or invalid token.
EXAMPLE 4

This is an example of the subsequent request and response when changes have occurred.

>> Request <<

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

>> Response <<

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

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:

STATUS:NEEDS-ACTION

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.

10. 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].
11. IANA Considerations

11.1. 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

11.2. 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

11.3. Link Relation Registrations

The following link relation values have been added to the Reference Types Registry defined in Section 6.2.2 of [RFC8288]:
Table 1

<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>

12. 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
VPOLL: Consensus Scheduling Component for iCalendar
draft-ietf-calext-vpoll-03

Abstract

This specification introduces a new RFC5545 iCalendar component which allows for consensus scheduling, that is, voting on a number of alternative meeting or task alternatives.

Status of This Memo

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

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

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

This Internet-Draft will expire on 6 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
Table of Contents

1. Acknowledgements ........................................... 3
2. Introduction ................................................. 3
3. Terms and definitions ....................................... 4
   3.1. consensus scheduling .................................. 4
   3.2. active Vpoll ............................................ 5
   3.3. voter .................................................. 5
4. Simple Consensus Scheduling ................................ 5
   4.1. The VPOLL Component: An Overview .................... 5
   4.2. The VPOLL Alternative Choices: An Overview ......... 7
   4.3. VPOLL responses ....................................... 8
   4.4. VPOLL updates ........................................ 9
   4.5. VPOLL Completion ...................................... 11
   4.6. Other Responses ....................................... 11
5. iCalendar Extensions ......................................... 12
   5.1. Updated Participant Type Value ....................... 12
   5.2. Updated Relation Type Value .......................... 12
   5.3. Updated Status Value ................................. 13
   5.4. New Property Parameters .............................. 13
      5.4.1. Required ......................................... 13
      5.4.2. Stay-Informed .................................... 14
   5.5. New Properties ........................................ 14
      5.5.1. Accept-Response ................................ 14
      5.5.2. Poll-Completion ................................ 15
      5.5.3. Poll-Item-Id .................................... 16
      5.5.4. Poll-Mode ....................................... 16
      5.5.5. Poll-properties ................................ 17
      5.5.6. Poll-Winner ..................................... 17
      5.5.7. Reply-URL ....................................... 18
      5.5.8. Response ....................................... 19
   5.6. New Components ........................................ 19
      5.6.1. VPOLL Component ................................. 19
      5.6.2. VOTE Component .................................. 22
6. Poll Modes .................................................. 23
   6.1. POLL-MODE:BASIC .................................... 24
      6.1.1. Property restrictions ............................ 24
      6.1.2. Outcome reporting ............................... 24
7. iTIP Extensions .............................................. 24
   7.1. Methods ............................................... 24
   7.2. Interoperability Models ............................... 26
      7.2.1. Delegation ....................................... 26
      7.2.2. Acting on Behalf of Other Calendar Users ....... 26
      7.2.3. Component Revisions ............................. 26
      7.2.4. Message Sequencing ................................ 26
   7.3. Application Protocol Elements ......................... 26
      7.3.1. Methods for VPOLL Calendar Components ........... 26
      7.3.2. Method: PUBLISH .................................. 28
1. Acknowledgements

The authors would like to thank the members of the Calendaring and Scheduling Consortium (CalConnect) for contributing their ideas and support, in particular Cyrus Daboo for his help with the specification and implementations.

2. Introduction

The currently existing approach to agreeing on meeting times using iTip [RFC5546] and/or iMip [RFC6047] has some significant failings. There is no useful bargaining or suggestion mechanism in iTip, only the ability for a potential attendee to accept or refuse or to counter with a time of their own choosing.
Part of the problem is that for many potential attendees, their freebusy is not an accurate representation of their availability. In fact, when trying to schedule conference calls across different organizations, attendees may not be allowed to provide freebusy information or availability as this may reveal something of the organizations internal activities.

A number of studies have shown that large amounts of time are spent trying to come to an agreement - up to and beyond 20 working hours per meeting. Many organizers fall back on other approaches such as phone calls and email to determine a suitable time.

Online services have appeared as a result and these allow participants to vote on a number of alternatives without revealing or using freebusy or availability. When agreement is reached a conventional scheduling message may be sent to the attendees. This approach appears to reach consensus fairly rapidly. Peer pressure may have some bearing on this as all voters are usually able to see the current state of the voting and may adjust their own meeting schedules to make themselves available for a popular choice.

The component and properties defined in this specification provide a standardized structure for this process and allow calendar clients and servers and web based services to interact.

These structures also have uses beyond the relatively simple needs of most meeting organizers. The process of coming to consensus can also be viewed as a bidding process.

3. Terms and definitions

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 [RFC2119].

The notation used in this memo to (re-)define iCalendar elements is the ABNF notation of [RFC5234] as used by [RFC5545]. Any syntax elements shown below that are not explicitly defined in this specification come from iCalendar [RFC5545].

Additionally the following terms are used:

3.1. consensus scheduling

The process whereby users come to some agreement on meeting or task alternatives and then book that meeting or task.
3.2. active Vpoll

A VPoll may have a DTSTART, DTEND and DURATION which may define the start and end of the active voting period.

3.3. voter

A participant who votes on the alternatives. A voter need not be an attendee of any of the alternatives presented.

4. Simple Consensus Scheduling

This specification defines components and properties which can be used for simple consensus scheduling but also have the generality to handle more complex cases. To provide an easy (and for many a sufficient) introduction to consensus scheduling and VPOLL we will outline the flow of information for the simple case of voting on a number of meeting alternatives which differ only in time of the meeting. In addition the voters will all be potential attendees.

This specification not only defines data structures but adds new iTip methods, one used when consensus has been reached and one to distribute the current status of the poll.

This document will show how a VPOLL object is used to inform voters of the state of a simple vote on some alternatives.

4.1. The VPOLL Component: An Overview

The VPOLL component acts as a wrapper for a number of alternatives to be voted on, together with some properties and a new component used to maintain the state of the voting. For our simple example the following VPOLL properties and sub-components are either required or appropriate:

DTSTAMP The usual [RFC5545] property.

SEQUENCE The usual [RFC5545] property. See below for SEQUENCE behavior.

UID The usual [RFC5545] property.

ORGANIZER The usual [RFC5545] property. In general this need not be an organizer of any of the alternatives. In this simple outline we assume it is the same.

SUMMARY The usual [RFC5545] property. This optional but recommended property provides the a short title to the poll.
DESCRIPTION The usual [RFC5545] property. This optional property provides more details.

DTEND The usual [RFC5545] property. This optional property provides a poll closing time and date after which the VPOLL is no longer active.

POLL-MODE A new property which defines how the votes are used to obtain a result. For our use case it will take the value "BASIC" meaning one event will be chosen from the alternatives.

POLL-COMPLETION A new property which defines who (server or client) chooses and/or submits the winning choice. In our example the value is "SERVER-SUBMIT" which means the client chooses the winner but the server will submit the winning choice.

POLL-PROPERTIES A new property which defines which icalendar properties are being voted on. For our use case it will take the value "DTSTART, LOCATION" meaning only those properties are significant for voting. Other properties in the events may differ but are not considered significant for the voting process.

PARTICIPANT There is one of these components for each voter with the PARTICIPANT-TYPE set to "VOTER". The CALENDAR-ADDRESS property identifies the voter and this component will contain one VOTE component for each item being voted on.

VOTE A new component. There is one of these for each voter and choice. It usually contains at least a POLL-ITEM-ID property to identify the choice and a RESPONSE property to provide a vote. For more complex poll modes it may contain other information such as cost or estimated duration.

VEVENT In our simple use case there will be multiple VEVENT sub-components defining the alternatives. Each will have a different date and or time for the meeting.

EXAMPLE

VPOLL with 3 voters and 3 alternative meetings:
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD:REQUEST
BEGIN:VPOLL
POLL-MODE:BASIC
POLL-COMPLETION:SERVER-SUBMIT
POLL-PROPERTIES:DTSTART,LOCATION
ORGANIZER:mailto:mike@example.com
UID:sched01-1234567890
DTSTAMP:20120101T000000Z
SUMMARY:What to do this week
DTEND:20120101T000000Z
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:cyrus@example.com
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:eric@example.com
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:mike@example.com
END:PARTICIPANT
BEGIN:VEVENT.......(with a poll-item-id=1)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=2)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=3)
END:VEVENT
END:VPOLL
END:VCALENDAR

As can be seen in the example above, there is an iTip METHOD property with the value REQUEST. The VPOLL object will be distributed to all the voters, either through iMip or through some VPOLL enabled service.

4.2. The VPOLL Alternative Choices: An Overview

Within the VPOLL component we have the alternatives to vote on. In many respects these are standard [RFC5545] components. For our simple use case they are all VEVENT components. In addition to the usual [RFC5545] properties some extra properties are used for a VPOLL.

POLL-ITEM-ID  This provides a unique reference to the sub-component
4.3. VPOLL responses

Upon receipt of a VPOLL REQUEST the voter will reply with a VPOLL component containing their vote. In our simple case it will have the following properties and components:

- **DTSTAMP** The usual [RFC5545] property.
- **SEQUENCE** The usual [RFC5545] property. See below for SEQUENCE behavior.
- **UID** Same as the request.
- **ORGANIZER** Same as the request.
- **SUMMARY** Same as the request.
- **PARTICIPANT** One only with a CALENDAR-ADDRESS identifying the voter replying.
- **VOTE** One per item being voted on.
- **POLL-ITEM-ID** One inside each VOTE component to identify the choice.
- **RESPONSE** One inside each VOTE component to specify the vote.

Note that a voter can send a number of REPLYS for each REQUEST sent by the organizer. In BASIC mode each REPLY completely replaces the voting record for that voter for all components being voted on. In our example, if Eric responds and votes for items 1 and 2 and then responds again with a vote for only item 3, the final outcome is one vote on item 3.

**NOTE** This is poll-mode specific behavior.

**EXAMPLE**

REPLY VPOLL from Cyrus:
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD: REPLY
BEGIN:VPOLL
ORGANIZER:mailto:mike@example.com
UID:sched01-1234567890
DTSTAMP:20120101T010000Z
SUMMARY:What to do this week
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:cyrus@example.com
BEGIN:VOTE
POLL-ITEM-ID:1
RESPONSE:50
COMMENT:Work on iTIP
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:2
RESPONSE:100
COMMENT:Work on WebDAV
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:3
RESPONSE:0
END:VOTE
END:PARTICIPANT
END:VPOLL
END:VCALENDAR

4.4.  VPOLL updates

When the organizer receives a response from one or more voters the current state of the poll is sent to all voters. The new iTip method POLLSTATUS is used. The VPOLL can contain a reduced set of properties but MUST contain DTSTAMP, SEQUENCE (if not 0), UID, ORGANIZER and one or more PARTICIPANT components each populated with zero or more VOTE components.

EXAMPLE
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD: POLLSTATUS
BEGIN:VPOLL
ORGANIZER:mailto:mike@example.com
UID:sched01-1234567890
DTSTAMP:20120101T020000Z
SEQUENCE:0
SUMMARY:What to do this week
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:cyrus@example.com
BEGIN:VOTE
POLL-ITEM-ID:1
RESPONSE:50
COMMENT:Work on iTIP
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:2
RESPONSE:100
COMMENT:Work on WebDAV
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:3
RESPONSE:0
END:VOTE
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:eric@example.com
BEGIN:VOTE
POLL-ITEM-ID:1
RESPONSE:100
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:2
RESPONSE:100
END:VOTE
BEGIN:VOTE
POLL-ITEM-ID:3
RESPONSE:0
END:VOTE
END:PARTICIPANT
END:VPOLL
END:VCALENDAR
4.5. VPOLL Completion

After a number of REPLY messages have been received the poll will be considered complete. If there is a DTEND on the poll the system may automatically close the poll, or the organizer may, at any time, consider the poll complete. A VPOLL can be completed (and effectively closed for voting) by sending an iTip REQUEST message with the VPOLL STATUS property set to COMPLETED.

The poll winner is confirmed by sending a final iTip REQUEST message with the VPOLL STATUS property set to CONFIRMED. In this case the VPOLL component contains all the events being voted on along with a POLL-WINNER property to identify the winning event. As the POLL-COMPLETION property is set to SERVER-SUBMIT the server will submit the winning choice and when it has done so set the STATUS to "SUBMITTED".

EXAMPLE

VPOLL confirmation:

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example//Example
METHOD: REQUEST
BEGIN:VPOLL
ORGANIZER:mailto:douglm@example.com
UID:sched01-1234567890
DTSTAMP:20120101T030000Z
COMPLETED:20120101T030000Z
POLL-COMPLETION:SERVER-SUBMIT
SEQUENCE:0
SUMMARY:What to do this week
STATUS:CONFIRMED
POLL-WINNER:3
BEGIN:VEVENT.......(with a poll-item-id=1)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=2)
END:VEVENT
BEGIN:VEVENT.......(with a poll-item-id=3)
END:VEVENT
END:VPOLL
END:VCALENDAR

4.6. Other Responses

A voter being asked to choose between a number of ORGANIZER supplied alternatives may find none of them acceptable or may simply not care.
An alternative response, which may be disallowed by the ORGANIZER, is to send back the respondees availability or freebusy or even one or more new, alternative choices.

This is accomplished by responding with a VOTE component which has no POLL-ITEM-ID property. In this case it MUST contain some alternative information. What form this takes depends on the poll mode in effect.

5. iCalendar Extensions

5.1. Updated Participant Type Value

Participant type property values are defined in section 11.2.1. of [RFC9073]. This specification updates that type to include the new participant type VOTER to provide information about the voter and to contain their votes.

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

```
partvalue /= "VOTER"
```

Description The new property value indicates that the associated PARTICIPANT component identifies a voter in a VPOLL.

5.2. Updated Relation Type Value

Relationship parameter type values are defined in section 3.2.15. of [RFC5545]. This specification updates that type to include the new relationship value POLL to provide a link to the VPOLL component in which the current component appears.

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

```
reltypeparam /= "RELTYPE" "=" "POLL"
```

Description This parameter can be specified on a property that references another related calendar component. The new parameter value indicates that the associated property references a VPOLL component which contains the current component.
5.3. Updated Status Value

Status property values are defined in section 3.8.1.11. of [RFC5545]. This specification updates that type to define valid VPOLL status values.

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

statvalue /= statvalue-poll
    ; Status values for "VPOLL".
statvalue-poll = "IN-PROCESS"
    / "COMPLETED"  ; Poll has closed, nothing has been chosen yet
    / "CONFIRMED"  ; Poll has closed and winning items confirmed
    / "SUBMITTED"  ; The winning item has been submitted
    / "CANCELLED"

Description  These values allow clients and servers to handle the choosing and submission of winning choices.

If the client is choosing and the server submitting then the client should set the POLL-WINNER property, set the status to CONFIRMED and save the poll. When the server submits the winning choice it will set the status to SUBMITTED.

Figure 1

5.4. New Property Parameters

5.4.1. Required

Parameter name  REQUIRED

Purpose  To specify whether the associated property is required in the current context.

Format Definition  This parameter is defined by the following notation:

requirededparam = "REQUIRED"  "="  ("TRUE" / "FALSE")
    ; Default is FALSE

Description  This parameter MAY be specified on REPLY-URL and, if the value is TRUE, indicates the organizer requires all replies to be made via the specified service rather than iTip replies.
5.4.2. Stay-Informed

Parameter name STAY-INFORMED

Purpose To specify the voter also wants to be added as an ATTENDEE when the poll is confirmed.

Format Definition This parameter is defined by the following notation:

stayinformedparam = "STAY-INFORMED" "=" ("TRUE" / "FALSE")
; Default is FALSE

Description This parameter MAY be specified on the CALENDAR-ADDRESS property in the PARTICIPANT component and, if the value is TRUE, indicates the voter wishes to be added to the final choice as a non participant.

5.5. New Properties

5.5.1. Accept-Response

Property name ACCEPT-RESPONSE

Purpose This property is used in VPOLL to indicate the types of component that may be supplied in a response.

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

Conformance This property MAY be specified in a VPOLL component.

Description When used in a VPOLL this property indicates what allowable component types may be returned in a reply. Typically this would allow a voter to respond with their freebusy or availability rather than choosing one of the presented alternatives.

If this property is not present voters are only allowed to respond to the choices in the request.

Format Definition This property is defined by the following notation:

acceptresponse = "ACCEPT-RESPONSE" acceptresponseparams "::
  iana-token ("," iana-token) CRLF
acceptresponseparams = *(";" other-param)
5.5.2. Poll-Completion

Property name POLL-COMPLETION

Purpose This property is used in VPOL to indicate whether the client or server is responsible for choosing and/or submitting the winner(s).

Description When a VPOL is stored on a server which is capable of handling choosing and submission of winning choices a value of SERVER indicates that the server should close the poll, choose the winner and submit whenever it is appropriate to do so.

For example, in BASIC poll-mode, reaching the DTEND of the poll could trigger this server side action.

Server initiated submission requires that the submitted choice MUST be a valid calendaring component.

POLL-COMPLETION=SERVER-SUBMIT allows the client to set the poll-winner, set the status to CONFIRMED and then store the poll on the server. The server will then submit the winning choice and set the status to SUBMITTED.

Format Definition This property is defined by the following notation:

poll-completion = "POLL-COMPLETION" pcparam ":" pcvalue CRLF

pcparam = *(";" other-param)

pcvalue = "SERVER" ; The server is responsible for both choosing and submitting the winner(s)
/ "SERVER-SUBMIT" ; The server is responsible for submitting the winner(s). The client chooses.
/ "SERVER-CHOICE" ; The server is responsible for choosing the winner(s). The client will submit.
/ "CLIENT" ; The client is responsible for both choosing and submitting the winner(s)
/ iana-token
/ x-name
;Default is CLIENT

Example The following is an example of this property:

POLL-COMPLETION: SERVER-SUBMIT
5.5.3. Poll-Item-Id

Property name  POLL-ITEM-ID

Purpose This property is used inVPOLL child components as an identifier.

Value type  INTEGER

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

Conformance  This property MUST be specified in a VOTE component and in VPOLL choice items.

Description  In a METHOD:REQUEST each choice component MUST have a POLL-ITEM-ID property. Each set of components with the same POLL-ITEM-ID value represents one overall set of items to be voted on.

POLL-ITEM-ID SHOULD be a unique small integer for each component or set of components. If it remains the same between REQUESTs then the previous response for that component MAY be re-used. To force a re-vote on a component due to a significant change, the POLL-ITEM-ID MUST change.

Format Definition  This property is defined by the following notation:

pollitemid = "POLL-ITEM-ID" pollitemdparams ":" integer CRLF

pollitemdparams = *( ";" other-param )

5.5.4. Poll-Mode

Property name  POLL-MODE

Purpose This property is used in VPOLL to indicate what voting mode is to be applied.

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

Conformance  This property MAY be specified in a VPOLL component or its sub-components.
Description The poll mode defines how the votes are applied to obtain a result. BASIC mode, the default, means that the voters are selecting one component (or group of components) with a given POLL=ITEM-ID.

Other polling modes may be defined in updates to this specification. These may allow for such modes as ranking or task assignment.

Format Definition This property is defined by the following notation:

pollmode = "POLL-MODE" pollmodeparams "：“
             ("BASIC" / iana-token / other-token) CRLF

pollmodeparams = *(";" other-param)

5.5.5. Poll-properties

Property name POLL-PROPERTIES

Purpose This property is used in VPOLL to define which icalendar properties are being voted on.

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

Conformance This property MAY be specified in a VPOLL component.

Description This property defines which icalendar properties are significant in the voting process. It may not be clear to voters which properties are varying in a significant manner. Clients may use this property to highlight those listed properties.

Format Definition This property is defined by the following notation:

pollproperties = "POLL-PROPERTIES" pollpropparams "：“
                 text *(""," text) CRLF

pollpropparams = *(";" other-param)

5.5.6. Poll-Winner

Property name POLL-WINNER

Purpose This property is used in a basic mode VPOLL to indicate which of the VPOLL sub-components won.
Value type  INTEGER

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

Conformance  This property MAY be specified in a VPOLL component.

Description  For poll confirmation each child component MUST have a POLL-ITEM-ID property. For basic mode the VPOLL component SHOULD have a POLL-WINNER property which MUST correspond to one of the POLL-ITEM-ID properties and indicates which sub-component was the winner.

Format Definition  This property is defined by the following notation:

    pollwinner = "POLL-WINNER" pollwinnerparams ":"
        integer CRLF

    pollwinnerparams = *(";" other-param)

        ; Used with a STATUS:CONFIRMED VPOLL to indicate which components have been confirmed

5.5.7.  Reply-URL

Property name  REPLY-URL

Purpose  This property may be used in scheduling messages to indicate additional reply methods, for example a web-service.

Value type  URI

Property Parameters  Non-standard, required or iana parameters can be specified on this property.

Conformance  This property MAY be specified in a VPOLL component.

Description  When used in a scheduling message this property indicates additional or required services that can be used to reply. Typically this would be a web service of some form.

Format Definition  This property is defined by the following notation:
reply-url = "REPLY-URL" reply-urlparams ":" uri CRLF

reply-urlparams = *( ";" requiredparam) /
                   (";" other-param)

5.5.8. Response

Property name RESPONSE

Purpose To specify a response vote.

Value type INTEGER

Format Definition This property is defined by the following notation:

response = "RESPONSE" response-params ":" integer CRLF
           ; integer value 0..100

responseparams = *(";" other-param)

Description This parameter can be specified on the POLL-ITEM-ID property to provide the value of the voters response. This parameter allows for fine grained responses which are appropriate to some applications. For the case of individuals voting for a choice of events, client applications SHOULD conform to the following convention:

* 0 - 39 A "NO vote"
* 40 - 79 A "MAYBE" vote
* 80 - 89 A "YES - but not preferred vote"
* 90-100 A "YES" vote.

Clients MUST preserve the response value when there is no change from the user even if they have a UI with fixed states (e.g. yes/no/maybe).

5.6. New Components

5.6.1. VPOLL Component

Component name VPOLL
Purpose
This component provides a mechanism by which voters can vote on provided choices.

Format Definition
This property is defined by the following notation:
Description This component provides a mechanism by which voters can vote on provided choices. The outcome depends upon the POLL-MODE in effect.
The PARTICIPANT components in VPOLL requests provide information on each recipient who will be voting - both their identity through the CALENDAR-ADDRESS property and their votes through the VOTE components.

If specified, the "DTSTART" property defines the start or opening of the poll active period. If absent the poll is presumed to have started when created.

If "DTSTART" is present "DURATION" MAY be specified and indicates the duration, and hence the ending, of the poll. The value of the property MUST be a positive duration.

"DTEND" MAY be specified with or without "DTSTART" and indicates the ending of the poll. If DTEND is specified it MUST be later than the DTSTART or CREATED property.

If one or more VALARM components are included in the VPOLL they are not components to be voted on and MUST NOT contain a POLL-ITEM-ID property. VALARM sub-components may be used to provide warnings to the user when polls are due to start or end.

5.6.2. VOTE Component

Component name VOTE

Purpose This component provides a mechanism by which voters can vote on provided choices.

Conformance This component may be specified zero or more times in a PARTICIPANT component which identifies the voter.

Format Definition This property is defined by the following notation:
votec = "BEGIN " : " VOTE" CRLF
  voteprop
  *eventc *todoc *journalc *freebusyc
  *availabilityc *alarmc *iana-comp *x-comp
  "END" ":" " VOTE" CRLF
voteprop = *{

  ; The following are REQUIRED,
  ; but MUST NOT occur more than once.
  ; pollitemid / response /
  ; The following are OPTIONAL,
  ; and MAY occur more than once.
  ; comment / x-prop / iana-prop
}

Description This component appears inside the PARTICIPANT component with a PARTICIPANT-TYPE of VOTER to identify the voter. This component contains that participants responses.

The required and optional properties and their meanings will depend upon the POLL-MODE in effect.

For any POLL-MODE, POLL-ITEM-ID is used to associate the information to a choice supplied by the organizer. This means that each VOTE component only provides information about that choice.

If allowed by the POLL-MODE a VOTE component without a POLL-ITEM-ID may be provided in a REPLY to indicate a possible new choice or to provide information to the ORGANIZER - such as the respondees availability.

6. Poll Modes

The VPOLL component is intended to allow for various forms of polling. The particular form in effect is indicated by the POLL-MODE property.

New poll modes can be registered by including a completed POLL-MODE Registration Template (see Section 10.3) in a published RFC.
6.1. POLL-MODE:BASIC

BASIC poll mode is the form of voting in which one possible outcome is chosen from a set of possibilities. Usually this will be represented as a number of possible event objects one of which will be selected.

6.1.1. Property restrictions

This poll mode has the following property requirements:

POLL-ITEM-ID  Each contained sub-component that is being voted upon MUST contain a POLL-ITEM_ID property which is unique within the context of the POLL. The value MUST NOT be reused when events are removed and/or added to the poll.

POLL-WINNER  On confirmation of the poll this property MUST be present and identifies the winning component.

6.1.2. Outcome reporting

To confirm the winner the POLL-WINNER property MUST be present and the STATUS MUST be set to CONFIRMED.

When the winning VEVENT or VTODO is not a scheduled entity, that is, it has no ORGANIZER or ATTENDEES it MUST be assigned an ORGANIZER property and a list of non-participating ATTENDEES. This allows the winning entity to be distributed to the participants through iTip or some other protocol.

7. iTIP Extensions

This specification introduces a number of extensions to [RFC5546]. In group scheduling the parties involved are organizer and attendees. In VPOLL the parties are organizer and voters.

For many of the iTip processing rules the voters take the place of attendees.

7.1. Methods

There are some extensions to the behavior of iTip methods for a VPOLL object and two new methods are defined.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLISH</td>
<td>No changes (yet)</td>
</tr>
</tbody>
</table>

REQUEST
Each child component MUST have a POLL-ITEM-ID property. Each set of components with the same POLL-ITEM-ID value represents one overall set of items to be voted on.

REPLY
There MUST be a single VPOLL component which MUST have: either one or more POLL-ITEM-ID properties with a RESPONSE param matching that from a REQUEST or a VFREEBUSY or VAVAILABILITY child component showing overall busy/available time. The VPOLL MUST have one voter only.

ADD
Not supported for VPOLL.

CANCEL
There MUST be a single VPOLL component with UID matching that of the poll being cancelled.

REFRESH
The organizer returns a METHOD:REQUEST with the current full state, or a METHOD:CANCEL or an error if no matching poll is found.

COUNTER
Not supported for VPOLL.

DECLINECOUNTER
Not supported for VPOLL.

POLLSTATUS
Used to send the current state of the poll to all voters. The VPOLL can contain a reduced set of properties but MUST contain DTSTAMP, SEQUENCE (if not 0), UID, ORGANIZER and PARTICIPANTS.

Table 1

The following table shows the above methods broken down by who can send them with VPOLL components.
### Table 2

<table>
<thead>
<tr>
<th>Originator</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizer</td>
<td>CANCEL, PUBLISH, REQUEST, POLLSTATUS</td>
</tr>
<tr>
<td>Voter</td>
<td>REPLY, REFRESH, REQUEST (only when delegating)</td>
</tr>
</tbody>
</table>

#### 7.2. Interoperability Models

Most of the standard iTip specification applies with respect to organizer and voters.

#### 7.2.1. Delegation

TBD

#### 7.2.2. Acting on Behalf of Other Calendar Users

TBD

#### 7.2.3. Component Revisions

* Need to talk about what a change in SEQUENCE means
* Sequence change forces a revote.
* New voter - no sequence change
* Add another poll set or change poll item ids or any change to a child
* component - bump sequence

#### 7.2.4. Message Sequencing

TBD

#### 7.3. Application Protocol Elements

#### 7.3.1. Methods for VPOLL Calendar Components

This section defines the property set restrictions for the method types that are applicable to the "VPOLL" calendar component. Each method is defined using a table that clarifies the property constraints that define the particular method.
The presence column uses the following values to assert whether a property is required or optional, and the number of times it may appear in the iCalendar object.

<table>
<thead>
<tr>
<th>Presence Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>One instance MUST be present.</td>
</tr>
<tr>
<td>1+</td>
<td>At least one instance MUST be present.</td>
</tr>
<tr>
<td>0</td>
<td>Instances of this property MUST NOT be present.</td>
</tr>
<tr>
<td>0+</td>
<td>Multiple instances MAY be present.</td>
</tr>
<tr>
<td>0 or 1</td>
<td>Up to 1 instance of this property MAY be present.</td>
</tr>
</tbody>
</table>

Table 3

The following summarizes the methods that are defined for the "VPOLL" calendar component.
### Table 4

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLISH</td>
<td>Post notification of a poll. Used primarily as a method of advertising the existence of a poll.</td>
</tr>
<tr>
<td>REQUEST</td>
<td>To make a request for a poll. This is an explicit invitation to one or more voters. Poll requests are also used to update, change or confirm an existing poll. Clients that cannot handle REQUEST MAY degrade the poll to view it as a PUBLISH. REQUEST SHOULD NOT be used just to set the status of the poll - POLLSTATUS provides a more compact approach.</td>
</tr>
<tr>
<td>REPLY</td>
<td>Reply to a poll request. Voters may set their RESPONSE parameter to supply the current vote in the range 0 to 100.</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Cancel a poll.</td>
</tr>
<tr>
<td>REFRESH</td>
<td>A request is sent to an Organizer by a Voter asking for the latest version of a poll to be resent to the requester.</td>
</tr>
<tr>
<td>POLLSTATUS</td>
<td>Used to send the current state of the poll to all voters. The VPOLL can contain a reduced set of properties but MUST contain DTSTAMP, SEQUENCE (if not 0), UID, ORGANIZER and PARTICIPANT.</td>
</tr>
</tbody>
</table>

7.3.2. Method: PUBLISH

The "PUBLISH" method in a "VPOLL" calendar component is an unsolicited posting of an iCalendar object. Any CU may add published components to their calendar. The "Organizer" MUST be present in a published iCalendar component. "Voters" MUST NOT be present. Its expected usage is for encapsulating an arbitrary poll as an iCalendar object. The "Organizer" may subsequently update (with another "PUBLISH" method) or cancel (with a "CANCEL" method) a previously published "VPOLL" calendar component.

Note: Not clear how useful this is but needs some work on transmitting the current vote without any voter identification.
This method type is an iCalendar object that conforms to the following property constraints:

<table>
<thead>
<tr>
<th>Component/Property</th>
<th>Presence</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1</td>
<td>MUST equal PUBLISH.</td>
</tr>
<tr>
<td>VPOOLL</td>
<td>1+</td>
<td></td>
</tr>
<tr>
<td>DTSTAMP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DTSTART</td>
<td>0 or 1</td>
<td>If present defines the start of the poll. Otherwise the poll starts when it is created and distributed.</td>
</tr>
<tr>
<td>ORGANIZER</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SUMMARY</td>
<td>1</td>
<td>Can be null.</td>
</tr>
<tr>
<td>UID</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>0 or 1</td>
<td>MUST be present if value is greater than 0; MAY be present if 0.</td>
</tr>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>ATTACH</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CATEGORIES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>COMPLETED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>CONTACT</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>CREATED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>0 or 1</td>
<td>Can be null.</td>
</tr>
<tr>
<td>DTEND</td>
<td>0 or 1</td>
<td>If present, DURATION MUST NOT be present.</td>
</tr>
<tr>
<td>DURATION</td>
<td>0 or 1</td>
<td>If present, DTEND MUST NOT be present.</td>
</tr>
<tr>
<td>Last-MODIFIED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>POLL-ITEM-ID</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>POLL-MODE</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>POLL-PROPERTIES</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>PRIORIY</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>RELATED-TO</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>RESOURCES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>IANA-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>X-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>PARTICIPANT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>REQUEST-STATUS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VALARM</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>VEVENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>VFREEBUSY</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VJOURNAL</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>VTODO</td>
<td>0+</td>
<td></td>
</tr>
</tbody>
</table>

PARTICIPANT components with PARTICIPANT-TYPE not equal to "VOTER" - that is, no voters

Depending upon the poll mode in effect there MAY be candidate components included in the poll component.

Depending upon the poll mode in effect there MAY be candidate components included in the poll component.

MAY be one of COMPLETED/CONFIRMED/CANCELLED.
7.3.3. Method: REQUEST

The "REQUEST" method in a "VPOLL" component provides the following scheduling functions:

* Invite "Voters" to respond to the poll.
* Change the items being voted upon.
* Complete or confirm the poll.
* Response to a "REFRESH" request.
* Update the details of an existing vpoll.
* Update the status of "Voters".
* Forward a "VPOLL" to another uninvited CU.
* For an existing "VPOLL" calendar component, delegate the role of "Voter" to another CU.
* For an existing "VPOLL" calendar component, change the role of "Organizer" to another CU.

The "Organizer" originates the "REQUEST". The recipients of the "REQUEST" method are the CUs voting in the poll, the "Voters". "Voters" use the "REPLY" method to convey votes to the "Organizer".

The "UID" and "SEQUENCE" properties are used to distinguish the various uses of the "REQUEST" method. If the "UID" property value in the "REQUEST" is not found on the recipient’s calendar, then the "REQUEST" is for a new "VPOLL" calendar component. If the "UID" property value is found on the recipient’s calendar, then the "REQUEST" is for an update, or a reconfirmation of the "VPOLL" calendar component.
For the "REQUEST" method only a single iCalendar object is permitted.

This method type is an iCalendar object that conforms to the following property constraints:

<table>
<thead>
<tr>
<th>Component/Property</th>
<th>Presence</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1</td>
<td>MUST be REQUEST.</td>
</tr>
<tr>
<td>VPOLL</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PARTICIPANT</td>
<td>1+</td>
<td>Identified as voters with the PARTICIPANT-TYPE=VOTER</td>
</tr>
<tr>
<td>DTSTAMP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DTSTART</td>
<td>0 or 1</td>
<td>If present defines the start of the poll. Otherwise the poll starts when it is created and distributed.</td>
</tr>
<tr>
<td>ORGANIZER</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>0 or 1</td>
<td>MUST be present if value is greater than 0; MAY be present if 0.</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>1</td>
<td>Can be null.</td>
</tr>
<tr>
<td>UID</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>ATTACH</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CATEGORIES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>COMPLETED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>CONTACT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CREATED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>0 or 1</td>
<td>Can be null.</td>
</tr>
</tbody>
</table>
DTEND 0 or 1 If present, DURATION MUST NOT be present.

DURATION 0 or 1 If present, DTEND MUST NOT be present.

GEO 0 or 1

LAST-MODIFIED 0 or 1

LOCATION 0 or 1

POLL-ITEM-ID 0

POLL-MODE 0 or 1

POLL-PROPERTIES 0 or 1

PRIORITY 0 or 1

RELATED-TO 0+

REQUEST-STATUS 0

RESOURCES 0+

STATUS 0 or 1 MAY be one of COMPLETED/CONFIRMED/CANCELLED.

TRANS 0 or 1

URL 0 or 1

IANA-PROPERTY 0+

X-PROPERTY 0+

VALARM 0+

VTIMEZONE 0+ MUST be present if any date/time refers to a timezone.

IANA-COMPONENT 0+

X-COMPONENT 0+

VEVENT 0+ Depending upon the poll mode in
Table 6: Constraints for a METHOD:REQUEST of a VPOLL

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFREEBUSY</td>
<td>MAY be candidate components included in the poll component.</td>
</tr>
<tr>
<td>VJOURNAL</td>
<td>Depending upon the poll mode in effect there MAY be candidate components included in the poll component.</td>
</tr>
<tr>
<td>VTODO</td>
<td>Depending upon the poll mode in effect there MAY be candidate components included in the poll component.</td>
</tr>
</tbody>
</table>

7.3.3.1. Rescheduling a poll

The "REQUEST" method may be used to reschedule a poll, that is force a revote. A rescheduled poll involves a change to the existing poll in terms of its time the components being voted on may have changed. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar but that the "SEQUENCE" (or "DTSTAMP") property value in the "REQUEST" method is greater than the value for the existing poll, then the "REQUEST" method describes a rescheduling of the poll.

7.3.3.2. Updating or Reconfirmation of a Poll

The "REQUEST" method may be used to update or reconfirm a poll. An update to an existing poll does not involve changes to the time or candidates, and might not involve a change to the location or description for the poll. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar and that the "SEQUENCE" property value in the "REQUEST" is the same as the value for the existing poll, then the "REQUEST" method describes an update of the poll details, but not a rescheduling of the POLL.

The update "REQUEST" method is the appropriate response to a "REFRESH" method sent from a "Voter" to the "Organizer" of a poll.
The "Organizer" of a poll may also send unsolicited "REQUEST" methods. The unsolicited "REQUEST" methods may be used to update the details of the poll without rescheduling it, to update the "RESPONSE" parameter of "Voters", or to reconfirm the poll.

7.3.3.3. Confirmation of a Poll

The "REQUEST" method may be used to confirm a poll, that is announce the winner in BASIC mode. The STATUS MUST be set to CONFIRMED and for BASIC mode a VPOLL POLL-WINNER property must be provided with the poll-id of the winning component.

7.3.3.4. Closing a Poll

The "REQUEST" method may be used to close a poll, that is indicate voting is completed. The STATUS MUST be set to COMPLETED.

7.3.3.5. Delegating a Poll to Another CU

Some calendar and scheduling systems allow "Voters" to delegate the vote to another "Calendar User". iTIP supports this concept using the following workflow. Any "Voter" may delegate their right to vote in a poll to another CU. The implication is that the delegate participates in lieu of the original "Voter", NOT in addition to the "Voter". The delegator MUST notify the "Organizer" of this action using the steps outlined below. Implementations may support or restrict delegation as they see fit. For instance, some implementations may restrict a delegate from delegating a "REQUEST" to another CU.

The "Delegator" of a poll forwards the existing "REQUEST" to the "Delegate". The "REQUEST" method MUST include a "Voter" property with the calendar address of the "Delegate". The "Delegator" MUST also send a "REPLY" method to the "Organizer" with the "Delegator’s" "Voter" property "DELEGATED-TO" parameter set to the calendar address of the "Delegate". Also, a new "Voter" property for the "Delegate" MUST be included and must specify the calendar user address set in the "DELEGATED-TO" parameter, as above.

In response to the request, the "Delegate" MUST send a "REPLY" method to the "Organizer", and optionally to the "Delegator". The "REPLY" method SHOULD include the "Voter" property with the "DELEGATED-FROM" parameter value of the "Delegator’s" calendar address.
The "Delegator" may continue to receive updates to the poll even though they will not be attending. This is accomplished by the "Delegator" setting their "role" attribute to "NON-PARTICIPANT" in the "REPLY" to the "Organizer".

### 7.3.3.6. Changing the Organizer

The situation may arise where the "Organizer" of a "VPOLL" is no longer able to perform the "Organizer" role and abdicates without passing on the "Organizer" role to someone else. When this occurs, the "Voters" of the "VPOLL" may use out-of-band mechanisms to communicate the situation and agree upon a new "Organizer". The new "Organizer" should then send out a new "REQUEST" with a modified version of the "VPOLL" in which the "SEQUENCE" number has been incremented and the "ORGANIZER" property has been changed to the new "Organizer".

### 7.3.3.7. Sending on Behalf of the Organizer

There are a number of scenarios that support the need for a "Calendar User" to act on behalf of the "Organizer" without explicit role changing. This might be the case if the CU designated as "Organizer" is sick or unable to perform duties associated with that function. In these cases, iTIP supports the notion of one CU acting on behalf of another. Using the "SENT-BY" parameter, a "Calendar User" could send an updated "VPOLL" "REQUEST". In the case where one CU sends on behalf of another CU, the "Voter" responses are still directed back towards the CU designated as "Organizer".

### 7.3.3.8. Forwarding to an Uninvited CU

A "Voter" invited to a "VPOLL" calendar component may send the "VPOLL" calendar component to another new CU not previously associated with the "VPOLL" calendar component. The current "Voter" participating in the "VPOLL" calendar component does this by forwarding the original "REQUEST" method to the new CU. The new CU can send a "REPLY" to the "Organizer" of the "VPOLL" calendar component. The reply contains a "Voter" property for the new CU.

The "Organizer" ultimately decides whether or not the new CU becomes part of the poll and is not obligated to do anything with a "REPLY" from a new (uninvited) CU. If the "Organizer" does not want the new CU to be part of the poll, the new "Voter" property is not added to the "VPOLL" calendar component. The "Organizer" MAY send the CU a "CANCEL" message to indicate that they will not be added to the poll.
If the "Organizer" decides to add the new CU, the new "Voter" property is added to the "VPOLL" calendar component. Furthermore, the "Organizer" is free to change any "Voter" property parameter from the values supplied by the new CU to something the "Organizer" considers appropriate. The "Organizer" SHOULD send the new CU a "REQUEST" message to inform them that they have been added.

When forwarding a "REQUEST" to another CU, the forwarding "Voter" MUST NOT make changes to the original message.

7.3.3.9. Updating Voter Status

The "Organizer" of an poll may also request updated status from one or more "Voters". The "Organizer" sends a "REQUEST" method to the "Voter" and sets the "RSVP=TRUE" property parameter on the PARTICIPANT CALENDAR-ADDRESS. The "SEQUENCE" property for the poll is not changed from its previous value. A recipient will determine that the only change in the "REQUEST" is that their "RSVP" property parameter indicates a request for updated status. The recipient SHOULD respond with a "REPLY" method indicating their current vote with respect to the "REQUEST".

7.3.4. Method: REPLY

The "REPLY" method in a "VPOLL" calendar component is used to respond (e.g., accept or decline) to a "REQUEST" or to reply to a delegation "REQUEST". When used to provide a delegation response, the "Delegator" SHOULD include the calendar address of the "Delegate" on the "DELEGATED-TO" property parameter of the "Delegator’s" "CALENDAR-ADDRESS" property. The "Delegate" SHOULD include the calendar address of the "Delegator" on the "DELEGATED-FROM" property parameter of the "Delegate’s" "CALENDAR-ADDRESS" property.

The "REPLY" method is also used when processing of a "REQUEST" fails. Depending on the value of the "REQUEST-STATUS" property, no action may have been performed.

The "Organizer" of a poll may receive the "REPLY" method from a CU not in the original "REQUEST". For example, a "REPLY" may be received from a "Delegate" to a poll. In addition, the "REPLY" method may be received from an unknown CU (a "Party Crasher"). This uninvited "Voter" may be accepted, or the "Organizer" may cancel the poll for the uninvited "Voter" by sending a "CANCEL" method to the uninvited "Voter".
A "Voter" MAY include a message to the "Organizer" using the "COMMENT" property. For example, if the user indicates a low interest and wants to let the "Organizer" know why, the reason can be expressed in the "COMMENT" property value.

The "Organizer" may also receive a "REPLY" from one CU on behalf of another. Like the scenario enumerated above for the "Organizer", "Voters" may have another CU respond on their behalf. This is done using the "SENT-BY" parameter.

The optional properties listed in the table below (those listed as "0+" or "0 or 1") MUST NOT be changed from those of the original request. (But see comments on VFREEBUSY and VAVAILABILITY)

This method type is an iCalendar object that conforms to the following property constraints:

<table>
<thead>
<tr>
<th>Component/Property</th>
<th>Presence</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1</td>
<td>MUST be REPLY.</td>
</tr>
<tr>
<td>VPOLL</td>
<td>1+</td>
<td>All components MUST have the same UID.</td>
</tr>
<tr>
<td>PARTICIPANT</td>
<td>1</td>
<td>Identifies the Voter replying.</td>
</tr>
<tr>
<td>DTSTAMP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ORGANIZER</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>UID</td>
<td>1</td>
<td>MUST be the UID of the original REQUEST.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>0 or 1</td>
<td>If non-zero, MUST be the sequence number of the original REQUEST. MAY be present if 0.</td>
</tr>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>ATTACH</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CATEGORIES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>COMPLETED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>CONTACT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CREATED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DTEND</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>If present, DURATION MUST NOT be present.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTSTART</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DURATION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>If present, DTEND MUST NOT be present.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEO</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>LAST-MODIFIED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>LOCATION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>POLL-ITEM-ID</td>
<td>1+</td>
<td></td>
</tr>
<tr>
<td>One per item being voted on.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POLL-MODE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>POLL-PROPERTIES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PRIORITY</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>RELATED-TO</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>RESOURCES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>REQUEST-STATUS</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>SUMMARY</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>TRANSP</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>IANA-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
</tbody>
</table>
7.3.5. Method: CANCEL

The "CANCEL" method in a "VPOLL" calendar component is used to send a cancellation notice of an existing poll request to the affected "Voters". The message is sent by the "Organizer" of the poll.

The "Organizer" MUST send a "CANCEL" message to each "Voter" affected by the cancellation. This can be done using a single "CANCEL" message for all "Voters" or by using multiple messages with different subsets of the affected "Voters" in each.

When a "VPOLL" is cancelled, the "SEQUENCE" property value MUST be incremented as described in Section 7.2.3.

Once a CANCEL message has been sent to all voters no further voting may take place. The poll is considered closed.
This method type is an iCalendar object that conforms to the following property constraints:

<table>
<thead>
<tr>
<th>Component/Property</th>
<th>Presence</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1</td>
<td>MUST be CANCEL.</td>
</tr>
<tr>
<td>VPOLL</td>
<td>1+</td>
<td>All must have the same UID.</td>
</tr>
<tr>
<td>PARTICIPANT</td>
<td>0+</td>
<td>MUST include some or all Voters being removed from the poll. MUST include some or all Voters if the entire poll is cancelled.</td>
</tr>
<tr>
<td>UID</td>
<td>1</td>
<td>MUST be the UID of the original REQUEST.</td>
</tr>
<tr>
<td>DTSTAMP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ORGANIZER</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ATTACH</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>COMPLETED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>CATEGORIES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>CONTACT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CREATED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DTEND</td>
<td>0 or 1</td>
<td>If present, DURATION MUST NOT be present.</td>
</tr>
<tr>
<td>DTSTART</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DURATION</td>
<td>0 or 1</td>
<td>If present, DTEND MUST NOT be present.</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>GEO</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>LAST-MODIFIED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>LOCATION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>POLL-ITEM-ID</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>POLL-MODE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>POLL-PROPERTIES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PRIORITY</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>RELATED-TO</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>RESOURCES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>0 or 1</td>
<td>MUST be set to CANCELLED to cancel the entire event. If uninviting specific Attendees, then MUST NOT be included.</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>TRANSP</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>IANA-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>X-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>REQUEST-STATUS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VALARM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VTIMEZONE</td>
<td>0+</td>
<td>MUST be present if any date/time refers to a timezone.</td>
</tr>
<tr>
<td>IANA-COMPONENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>X-COMPONENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>VTODO</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
7.3.6. Method: REFRESH

The "REFRESH" method in a "VPOLL" calendar component is used by "Voters" of an existing event to request an updated description from the poll "Organizer". The "REFRESH" method must specify the "UID" property of the poll to update. The "Organizer" responds with the latest description and version of the poll.

This method type is an iCalendar object that conforms to the following property constraints:

<table>
<thead>
<tr>
<th>Component/Property</th>
<th>Presence</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1</td>
<td>MUST be REFRESH.</td>
</tr>
<tr>
<td>VJOURNAL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VEVENT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VFREEBUSY</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VPOLL</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PARTICIPANT</td>
<td>1</td>
<td>MUST identify the requester as a voter.</td>
</tr>
<tr>
<td>DTSTAMP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ORGANIZER</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>UID</td>
<td>1</td>
<td>MUST be the UID associated with original REQUEST.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>COMPLETED</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IANA-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>X-PROPERTY</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Constraints for a METHOD:CANCEL of a VPOLL
<table>
<thead>
<tr>
<th>ATTACH</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORIES</td>
<td>0</td>
</tr>
<tr>
<td>CLASS</td>
<td>0</td>
</tr>
<tr>
<td>CONTACT</td>
<td>0</td>
</tr>
<tr>
<td>CREATED</td>
<td>0</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>0</td>
</tr>
<tr>
<td>DTEND</td>
<td>0</td>
</tr>
<tr>
<td>DTSTART</td>
<td>0</td>
</tr>
<tr>
<td>DURATION</td>
<td>0</td>
</tr>
<tr>
<td>GEO</td>
<td>0</td>
</tr>
<tr>
<td>LAST-MODIFIED</td>
<td>0</td>
</tr>
<tr>
<td>LOCATION</td>
<td>0</td>
</tr>
<tr>
<td>POLL-ITEM-ID</td>
<td>0</td>
</tr>
<tr>
<td>POLL-MODE</td>
<td>0</td>
</tr>
<tr>
<td>POLL-PROPERTIES</td>
<td>0</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>0</td>
</tr>
<tr>
<td>RELATED-TO</td>
<td>0</td>
</tr>
<tr>
<td>REQUEST-STATUS</td>
<td>0</td>
</tr>
<tr>
<td>RESOURCES</td>
<td>0</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>0</td>
</tr>
<tr>
<td>STATUS</td>
<td>0</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>0</td>
</tr>
<tr>
<td>URL</td>
<td>0</td>
</tr>
<tr>
<td>VALARM</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.3.7.  Method: POLLSTATUS

The "POLLSTATUS" method in a "VPOLL" calendar component is used to inform recipients of the current status of the poll in a compact manner. The "Organizer" MUST be present in the confirmed poll component. All "Voters" MUST be present. The selected component(s) according to the poll mode SHOULD NOT be present in the poll component. Clients receiving this message may store the confirmed items in their calendars.

This method type is an iCalendar object that conforms to the following property constraints:

<table>
<thead>
<tr>
<th>Component/Property</th>
<th>Presence</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>1</td>
<td>MUST equal POLLSTATUS.</td>
</tr>
<tr>
<td>VPOLL</td>
<td>1+</td>
<td></td>
</tr>
<tr>
<td>PARTICIPANT</td>
<td>1+</td>
<td>The voters containing their current vote</td>
</tr>
<tr>
<td>COMPLETED</td>
<td>0 or 1</td>
<td>Only present for a completed poll</td>
</tr>
<tr>
<td>DTSTAMP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DTSTART</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>ORGANIZER</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>UID</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ATTACH</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CATEGORIES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>CONTACT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CREATED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DTEND</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>DURATION</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>LAST-MODIFIED</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>POLL-ITEM-ID</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>POLL-MODE</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>POLL-PROPERTIES</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PRIORITY</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>RELATED-TO</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>RESOURCES</td>
<td>0+</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>0 or 1</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>0 or 1</td>
<td></td>
</tr>
</tbody>
</table>
8. CalDAV Extensions

This specification extends [RFC4791] in that it defines a new component and new iCalendar properties to be supported and requires extra definitions related to time-ranges and reports.

Additionally, it extends [RFC6638] as it a VPOLL component is a schedulable entity.

8.1. Calendar Collection Properties

This section defines new CalDAV properties for calendar collections.

8.1.1. CALDAV:supported-vpoll-component-sets

Name supported-vpoll-component-sets

Namespace urn:ietf:params:xml:ns:caldav
Purpose  Specifies the calendar component types (e.g., VEVENT, VTODO, etc.) and combination of types that may be included in a VPOLL component.

Conformance  This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in Section 12.14.1 of [RFC2518]).

Description  The CALDAV:supported-vpoll-component-sets property is used to specify restrictions on the calendar component types that VPOLL components may contain in a calendar collection.

It also specifies the combination of allowed component types.

Any attempt by the client to store VPOLL components with component types or combinations of types not listed in this property, if it exists, MUST result in an error, with the CALDAV:supported-vpoll-component-sets precondition Section 8.2 being violated. Since this property is protected, it cannot be changed by clients using a PROPPATCH request. However, clients can initialize the value of this property when creating a new calendar collection with MKCALENDAR. In the absence of this property, the server MUST accept all component types, and the client can assume that all component types are accepted.

Definition

<!ELEMENT supported-vpoll-component-sets
  (supported-vpoll-component-set*) >

<!ELEMENT supported-vpoll-component-set (comp)+>
<C:supported-vpoll-component-sets
  xmlns:C="urn:ietf:params:xml:ns:caldav">
  <!-- VPOLLs with VEVENT, VFREEBUSY or VTODO -->
  <C:supported-vpoll-component-set>
    <C:comp name="VEVENT" />
    <C:comp name="VFREEBUSY" />
    <C:comp name="VTODO" />
  </C:supported-vpoll-component-set>

  <!-- VPOLLs with just VEVENT or VFREEBUSY -->
  <C:supported-vpoll-component-set>
    <C:comp name="VEVENT" />
    <C:comp name="VFREEBUSY" />
  </C:supported-vpoll-component-set>

  <!-- VPOLLs with just VEVENT -->
  <C:supported-vpoll-component-set>
    <C:comp name="VEVENT" />
  </C:supported-vpoll-component-set>

  <!-- VPOLLs with just VTTODO -->
  <C:supported-vpoll-component-set>
    <C:comp name="VTTODO" />
  </C:supported-vpoll-component-set>
</C:supported-vpoll-component-sets>

8.1.2. CALDAV:vpoll-max-items

Name  vpoll-max-items

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

Purpose  Provides a numeric value indicating the maximum number of items that may be contained in any instance of a VPOLL calendar object resource stored in the calendar collection.

Conformance  This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in Section 12.14.1 of [RFC2518]).

Description  The CALDAV:vpoll-max-items is used to specify a numeric value that indicates the maximum number of iCalendar components in any one instance of a VPOLL calendar object resource stored in a calendar collection. Any attempt to store a calendar object resource with more components per instance than this value MUST result in an error, with the CALDAV: vpoll-max-items precondition
Section 8.2 being violated. In the absence of this property, the client can assume that the server can handle any number of items in a VPOLL calendar component.

Definition

<!ELEMENT vpoll-max-items (#PCDATA)>
PCDATA value: a numeric value (integer greater than zero)

<?xml-stylesheet type="text/xsl" href="normative-xsl.xsl"?>

8.1.3. CALDAV:vpoll-max-active

Name vpoll-max-active

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

Purpose Provides a numeric value indicating the maximum number of active vpolls at any one time.

Conformance This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in Section 12.14.1 of [RFC2518]).

Description The CALDAV:vpoll-max-active is used to specify a numeric value that indicates the maximum number of active VPOLLs at any one time. Any attempt to store a new active VPOLL calendar object resource which results in exceeding this limit MUST result in an error, with the CALDAV:vpoll-max-active precondition Section 8.2 being violated. In the absence of this property, the client can assume that the server can handle any number of active VPOLLs.

Definition

<!ELEMENT vpoll-max-active (#PCDATA)>
PCDATA value: a numeric value (integer greater than zero)

8.1.4. CALDAV:vpoll-max-voters

Name vpoll-max-voters

Namespace urn:ietf:params:xml:ns:caldav
Purpose  Provides a numeric value indicating the maximum number of voters for any instance of a VPOLL calendar object resource stored in the calendar collection.

Conformance  This property MAY be defined on any calendar collection. If defined, it MUST be protected and SHOULD NOT be returned by a PROPFIND DAV:allprop request (as defined in Section 12.14.1 of [RFC2518]).

Description  The CALDAV:vpoll-max-voters is used to specify a numeric value that indicates the maximum number of voters for any one instance of a VPOLL calendar object resource stored in a calendar collection. Any attempt to store a calendar object resource with more voters per instance than this value MUST result in an error, with the CALDAV: vpoll-max-voters precondition Section 8.2 being violated. In the absence of this property, the client can assume that the server can handle any number of voters in a VPOLL calendar component.

Definition

<!ELEMENT vpoll-max-voters (#PCDATA)>
PCDATA value: a numeric value (integer greater than zero)

<C:vpoll-max-voters xmlns:C="urn:ietf:params:xml:ns:caldav">
  25
</C:vpoll-max-voters>

8.1.5.  CalDAV:even-more-properties

8.1.6.  Extensions to CalDAV scheduling

This specification extends [RFC6638].

Each section of Appendix A "Scheduling Privileges Summary" is extended to include VPOLL.

Any reference to the ATTENDEE property should be read to include the CALENDAR-ADDRESS property contained in the PARTICIPANT components. That is, for scheduling purposes the CALENDAR-ADDRESS property is handled in exactly the same manner as the ATTENDEE property.
8.2. Additional Preconditions for PUT, COPY, and MOVE

This specification creates additional Preconditions for PUT, COPY, and MOVE methods. These preconditions apply when a PUT operation of a VPOLL calendar object resource into a calendar collection occurs, or when a COPY or MOVE operation of a calendar object resource into a calendar collection occurs, or when a COPY or MOVE operation occurs on a calendar collection.

The new preconditions are:

(CALDAV:supported-vpoll-component-sets) The VPOLL resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST contain a type or combination of calendar component that is supported in the targeted calendar collection;

(CALDAV:vpoll-max-items) The VPOLL resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have a number of sub-components (excluding VTIMEZONE) less than or equal to the value of the CALDAV:vpoll-max-items property value Section 8.1.2 on the calendar collection where the resource will be stored;

(CALDAV:vpoll-max-active) The PUT request, or COPY or MOVE request, MUST not result in the number of active VPOLLs being greater than the value of the CALDAV:vpoll-max-active property value Section 8.1.3 on the calendar collection where the resource will be stored;

(CALDAV:vpoll-max-voters) The VPOLL resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have a number of voters represented by PARTICIPANT components less than or equal to the value of the CALDAV:vpoll-max-voters property value Section 8.1.4 on the calendar collection where the resource will be stored;

8.3. CalDAV:calendar-query Report

This allows the retrieval of VPOLLs and their included components. The query specification allows queries to be directed at the contained sub-components. For VPOLL queries this feature is disallowed. Time-range queries can only target the vpoll component itself.
8.3.1. Example: Partial Retrieval of VPoll

In this example, the client requests the server to return specific components and properties of the VPoll components that overlap the time range from December 4, 2012, at 00:00:00 A.M. UTC to December 5, 2012, at 00:00:00 A.M. UTC. In addition, the DAV:getetag property is also requested and returned as part of the response. Note that due to the CALDAV: calendar-data element restrictions, the DTSTAMP property in VPoll components has not been returned, and the only property returned in the VCALENDAR object is VERSION.

>> Request <<

REPORT /cyrus/work/ HTTP/1.1
Host: cal.example.com
Depth: 1
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:calendar-query xmlns:D="DAV:"
 xmlns:C="urn:ietf:params:xml:ns:caldav">
 <D:prop>
  <D:getetag/>
  <C:calendar-data>
   <C:comp name="VCALENDAR">
    <C:comp name="VPOLL">
     <C:prop name="SUMMARY"/>
     <C:prop name="UID"/>
     <C:prop name="DTSTART"/>
     <C:prop name="DTEND"/>
     <C:prop name="DURATION"/>
    </C:comp>
   </C:comp>
  </C:calendar-data>
 </D:prop>
 <C:filter>
  <C:comp-filter name="VCALENDAR">
   <C:time-range start="20121204T000000Z"
                end="20121205T000000Z"/>
  </C:comp-filter>
 </C:filter>
</C:calendar-query>
HTTP/1.1 207 Multi-Status
Date: Sat, 11 Nov 2012 09:32:12 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:
xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:response>
    <D:href>http://cal.example.com/cyrus/work/poll2.ics</D:href>
    <D:propstat>
      <D:prop>
        <D:getetag>fffff-abcd2</D:getetag>
        <C:calendar-data>
          BEGIN:VCALENDAR
          VERSION:2.0
          BEGIN:VPOLL
          DTSTART;TZID=US/Eastern:20121202T120000
          DURATION:PT4D
          SUMMARY:Poll #2
          UID:00959BC664CA650E933C892C@example.com
          END:VPOLL
          END:VCALENDAR
        </C:calendar-data>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:propstat>
      <D:prop>
        <D:getetag>fffff-abcd3</D:getetag>
        <C:calendar-data>
          BEGIN:VCALENDAR
          VERSION:2.0
          PRODID:-//Example Corp.//CalDAV Client//EN
          BEGIN:VPOLL
          DTSTART;TZID=US/Eastern:20121204T100000
          DURATION:PT4D
          SUMMARY:Poll #3
          UID:DC6C50A017428C5216A2F1CD@example.com
          END:VPOLL
          END:VCALENDAR
        </C:calendar-data>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
8.4. CalDAV time ranges

"CALDAV:time-range XML Element" in Section 9.9 of [RFC4791] describes how to specify time ranges to limit the set of calendar components returned by the server. This specification extends [RFC4791] to describe the meaning of time ranges for VPOLL.

A VPOLL component is said to overlap a given time range if the condition for the corresponding component state specified in the table below is satisfied. The conditions depend on the presence of the DTSTART, DURATION, DTEND, COMPLETED and CREATED properties in the VPOLL component. Note that, as specified above, the DTEND value MUST be a DATE-TIME value equal to or after the DTSTART value if specified.
VPOLL has the DTSTART property?
- VPOLL has the DURATION property?
  - VPOLL has the DTEND property?
    - VPOLL has the COMPLETED property?
      - VPOLL has the CREATED property?
        Condition to evaluate

Y Y N * * (start <= DTSTART+DURATION) AND
  ((end > DTSTART) OR
   (end >= DTSTART+DURATION))

Y N Y * * ((start < DTEND) OR (start <= DTSTART))
  AND
  ((end > DTSTART) OR (end >= DTEND))

Y N N * * (start <= DTSTART) AND (end > DTEND)

N N Y * * (start < DTEND) AND (end >= DTEND)

N N N Y Y ((start <= CREATED) OR (start <= COMPLETED))
  AND
  ((end >= CREATED) OR (end >= COMPLETED))

N N N Y N (start <= COMPLETED) AND (end >= COMPLETED)

N N N N Y (end > CREATED)

N N N N N TRUE

9. Security Considerations

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

10. IANA Considerations

10.1. Parameter Registrations

This document defines the following new iCalendar property parameters
to be added to the registry defined in Section 8.2.4 of [RFC5545]:

York & Douglass Expires 6 September 2022 [Page 56]
10.2. Property Registrations

This document defines the following new iCalendar properties to be added to the registry defined in Section 8.2.3 of [RFC5545]:

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT-RESPONSE</td>
<td>Current</td>
<td>Section 5.5.7</td>
</tr>
<tr>
<td>POLL-ITEM-ID</td>
<td>Current</td>
<td>Section 5.5.3</td>
</tr>
<tr>
<td>POLL-MODE</td>
<td>Current</td>
<td>Section 5.5.4</td>
</tr>
<tr>
<td>POLL-PROPERTIES</td>
<td>Current</td>
<td>Section 5.5.5</td>
</tr>
<tr>
<td>POLL-WINNER</td>
<td>Current</td>
<td>Section 5.5.6</td>
</tr>
<tr>
<td>RESPONSE</td>
<td>Current</td>
<td>Section 5.5.8</td>
</tr>
</tbody>
</table>

Table 12

10.3. POLL-MODE Registration Template

A poll mode is defined by completing the following template.

Poll mode name  The name of the poll mode.

Purpose  The purpose of the poll mode. Give a short but clear description.

Reference  A reference to the RFC in which the poll mode is defined

10.4. POLL-MODE Registrations

This document defines the following registered poll modes.
<table>
<thead>
<tr>
<th>Poll mode name</th>
<th>Purpose</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC</td>
<td>To provide simple voting for a single outcome from a number of candidates.</td>
<td>Current</td>
</tr>
</tbody>
</table>

Table 13

11. Normative References


Internet-Draft                    VPOLL                       March 2022

Appendix A.  Open issues

public-comment: Not documented and was a parameter on something. Really sounds like a PARTICIPANT or VOTE property

Notifications: Need to do a section on what Notifications to support. A. VPOLL is about to end and you haven’t voted on it yet. Instead reuse VALARMS to notify the user?

Future: Restarting a confirmed/completed VPOLL What to do with changes to STATUS:CONFIRMED? Allow them or not? What do to that poll had a winning event or todo. Stress VPOLL UID MUST be unique Changing status back from CONFIRMED MUST adjust status of any events booked as a result of confirmation. MUST winning event be cancelled for POLL-MODE basic? No - voter has indicated now unable to attend - want to revote

Future: Voting on a confirmed/completed VPOLL Can a voter vote after completion? May be unable to attend and wants to indicate. Requires retention of VPOLL retention period Removed status

ORGANIZER/ATTENDEE validity Can a user create a poll with scheduled events where that user’s isn’t the organizer of the poll? So is there a requirement that the account that poll is on is able to create each one of the resources in the poll? i.e. I can’t create a poll with a set of events where I am just the attendee of the events. Are there any other restrictions for components in a VPOLL? Add to security consideration

Update to existing event after poll confirm When voting on existing event - winning properties ONLY are merged in to the real event.

Need to write down what isn’t valid in a VPOLL
a. Can’t change POLL-MODE
Guide for ATTENDEE roles chair, NON-PARTICIPANT etc

? - some iTip notes On confirm - send itip if appropriate (PUBLISH) - all non-participating - shared - feeds Organizer can specify where result is? Confirm can specify that itip is sent - ITIP / NONE - parameter ? on POLL-WINNER

Need to add example of freebusy in response

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//BedeworkCaldavTest//BedeworkCaldavTest
METHOD: REPLY
BEGIN:VPOLL
ORGANIZER:mailto:douglm@mysite.edu
BEGIN:PARTICIPANT
PARTICIPANT-TYPE: VOTER
CALENDAR-ADDRESS:mailto:eric@example.com
UID:sched01-1234567890
DTSTAMP:20120101T010000Z
SEQUENCE:0
SUMMARY:What to do this week
BEGIN:VFREEBUSY
........
END:VFREEBUSY
END:PARTICIPANT
END:VPOLL
END:VCALENDAR

Appendix B. Change log

Calext V01: 2019-10-17 MD Replace VVOTER and VOTER with PARTICIPANT.

Calext V00: 2019-05-17 MD First calext version. Moved source to metanorma. No changes to specification.

V03: 2014-10-28 MD
* Add VVOTER and VOTE components.
* Add RESPONSE property.
* Remove RESPONSE parameter from VOTER.

V03: 2014-05-12 MD
* Add reply-url property and required parameter.
* Fix ACCEPT-RESPONSE definition.
V02: 2014-05-12 MD
* Typos fixed, clarifications made.

* Removed spurious COMMENT param. Switched some to PUBLIC-COMMENT

* Changed STAY-INFORMED to remove boolean value type and state explicit TRUE/FALSE values.

* iTip: Allow VPOLL DTSTART to be optional and allow VAVAILABILITY as subcomponent

* iTip: fix broken table cells

* Add POLL-PROPERTIES, POLL-WINNER to 5545 extensions table

* Added Caldav scheduling section

V01: 2013-08-07 MD
* Removed method CONFIRM

* Removed pollitemid from VPOLL abnf. Added text for pollwinner

* Added POLL-WINNER and verbiage

* Added STATUS values

* Added RELTYPE=POLL

* Added supported-vpoll-component-sets

* Added CalDAV related parameters to VOTER

* Removed bad CalDAV query example. State that queries cannot target the sub-components.

Initial version: 2012-11-02 MD

Authors’ Addresses

Eric York
Email: eric.york@gmail.com

Michael Douglass
Email: mdouglass@bedework.com