

Network Working Group  
Internet-Draft  
Updates: 5545 (if approved)  
Intended status: Standards Track  
Expires: September 6, 2019

C. Daboo  
Apple  
K. Murchison, Ed.  
FastMail  
March 5, 2019

VALARM Extensions for iCalendar  
draft-daboo-valarm-extensions-05

Abstract

This document defines a set of extensions to the iCalendar VALARM component to enhance use of alarms and improve interoperability between clients and servers.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

# Table of Contents

1.	Introduction . . . . .	2
2.	Conventions Used in This Document . . . . .	3
3.	Extensible syntax for VALARM . . . . .	3
4.	Alarm Unique Identifier . . . . .	5
5.	Alarm Acknowledgement . . . . .	5
5.1.	Acknowledged Property . . . . .	6
6.	Snoozing Alarms . . . . .	7
7.	Alarm Proximity Trigger . . . . .	8
7.1.	Proximity Property . . . . .	9
7.2.	Example . . . . .	10
8.	Security Considerations . . . . .	10
9.	IANA Considerations . . . . .	10
9.1.	Property Registrations . . . . .	10
9.2.	Proximity Value Registry . . . . .	11
10.	Acknowledgments . . . . .	11
11.	References . . . . .	11
11.1.	Normative References . . . . .	11
11.2.	Informative References . . . . .	12
11.3.	URIs . . . . .	12
Appendix A. Change History (To be removed by RFC Editor before publication) . . . . .		12
Authors' Addresses . . . . .		14

## 1. Introduction

The iCalendar [RFC5545] specification defines a set of components used to describe calendar data. One of those is the "VALARM" component which appears as a sub-component of "VEVENT" and "VTODO" components. The "VALARM" component is used to specify a reminder for an event or task. Different alarm actions are possible, as are different ways to specify how the alarm is triggered.

As iCalendar has become more widely used and as client-server protocols such as CalDAV [RFC4791] have become more popular, several issues with "VALARM" components have arisen. Most of these relate to the need to extend the existing "VALARM" component with new properties and behaviors to allow clients and servers to accomplish specific tasks in an interoperable manner. For example, clients typically need a way to specify that an alarm has been dismissed by a calendar user, or has been "snoozed" by a set amount of time. To date, this has been done through the use of custom "X-" properties specific to each client implementation, leading to poor interoperability.

This specification defines a set of extensions to "VALARM" components to cover common requirements for alarms not currently addressed in

iCalendar. Each extension is defined in a separate section below. For the most part, each extension can be supported independently of the others, though in some cases one extension will require another. In addition, this specification describes mechanisms by which clients can interoperably implement common features such as "snoozing".

## 2. 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 [1] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

When XML element types in the namespaces "DAV:" and "urn:ietf:params:xml:ns:caldav" are referenced in this document outside of the context of an XML fragment, the string "DAV:" and "CALDAV:" will be prefixed to the element type names respectively.

## 3. Extensible syntax for VALARM

Section 3.6.6 of [RFC5545] defines the syntax for "VALARM" components and properties within them. However, as written, it is hard to extend this by adding, e.g., a new property common to all types of alarm. Since many of the extensions defined in this document need to extend the base syntax, an alternative form for the base syntax is defined here, with the goal of simplifying specification of the extensions.

A "VALARM" calendar component is re-defined by the following notation:

```
alarmcext = "BEGIN" ":" "VALARM" CRLF
           alarmprop
           "END" ":" "VALARM" CRLF

alarmprop = *(
    ; the following are REQUIRED,
    ; but MUST NOT occur more than once

    action / trigger /

    ; one set of action properties MUST be
    ; present and MUST match the action specified
    ; in the ACTION property

    actionprops /
```

```
        ; the following is OPTIONAL,  
        ; and MAY occur more than once  
  
        x-prop / iana-prop  
    )  
  
actionprops = audiopropext / disppropext / emailpropext  
audiopropext = *(  
    ; 'duration' and 'repeat' are both OPTIONAL,  
    ; and MUST NOT occur more than once each,  
    ; but if one occurs, so MUST the other  
  
    duration / repeat /  
  
    ; the following is OPTIONAL,  
    ; but MUST NOT occur more than once  
  
    attach  
    )  
disppropext = *(  
    ; the following are REQUIRED,  
    ; but MUST NOT occur more than once  
  
    description /  
  
    ; 'duration' and 'repeat' are both OPTIONAL,  
    ; and MUST NOT occur more than once each,  
    ; but if one occurs, so MUST the other  
  
    duration / repeat  
    )  
emailpropext = *(  
    ; the following are all REQUIRED,  
    ; but MUST NOT occur more than once  
  
    description / summary /  
  
    ; the following is REQUIRED,  
    ; and MAY occur more than once
```

```
attendee /  
  
; 'duration' and 'repeat' are both OPTIONAL,  
; and MUST NOT occur more than once each,  
; but if one occurs, so MUST the other  
  
duration / repeat  
  
)
```

#### 4. Alarm Unique Identifier

This extension adds a "UID" property to "VALARM" components to allow a unique identifier to be specified. The value of this property can then be used to refer uniquely to the "VALARM" component.

The "UID" property defined here follows the definition in Section 3.8.4.7 of [RFC5545] with the security and privacy updates in Section 5.3 of [RFC7986]. In particular it MUST be a globally unique identifier that does not contain any security- or privacy-sensitive information.

The "VALARM" component defined in Section 3 is extended here as:

```
alarmprop /= *(  
  
; the following is OPTIONAL,  
; but MUST NOT occur more than once  
  
uid  
  
)
```

#### 5. Alarm Acknowledgement

There is currently no way for a "VALARM" component to indicate whether it has been triggered and acknowledged. With the advent of a standard client/server protocol for calendaring and scheduling data ([RFC4791]) it is quite possible for an event with an alarm to exist on multiple clients in addition to the server. If each of those is responsible for performing the action when an alarm triggers, then multiple "alerts" are generated by different devices. In such a situation, a calendar user would like to be able to "dismiss" the alarm on one device and have it automatically dismissed on the others too.

Also, with recurring events that have alarms, it is important to know when the last alarm in the recurring set was acknowledged, so that the client can determine whether past alarms have been missed.

To address these needs, this specification adds an "ACKNOWLEDGED" property to "VALARM" components to indicate when the alarm was last sent or acknowledged. This is defined by the syntax below.

```
alarmprop      /= *(  
                ; the following is OPTIONAL,  
                ; but MUST NOT occur more than once  
  
                acknowledged  
  
                )
```

#### 5.1. Acknowledged Property

Property Name: ACKNOWLEDGED

Purpose: This property specifies the UTC date and time at which the corresponding alarm was last sent or acknowledged.

Value Type: DATE-TIME

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

Conformance: This property can be specified within "VALARM" calendar components.

Description: This property is used to specify when an alarm was last sent or acknowledged. This allows clients to determine when a pending alarm has been acknowledged by a calendar user so that any alerts can be dismissed across multiple devices. It also allows clients to track repeating alarms or alarms on recurring events or to-dos to ensure that the right number of missed alarms can be tracked.

Clients SHOULD set this property to the current date-time value in UTC when a calendar user acknowledges a pending alarm. Certain kinds of alarm may not provide feedback as to when the calendar user sees them, for example email based alerts. For those kinds of alarms, the client SHOULD set this property when the alarm is triggered and the action successfully carried out.

When an alarm is triggered on a client, clients can check to see if an "ACKNOWLEDGED" property is present. If it is, and the value of that property is greater than or equal to the computed trigger time for the alarm, then the client SHOULD NOT trigger the alarm. Similarly, if an alarm has been triggered and an "alert" presented to a calendar user, clients can monitor the iCalendar data to determine whether an "ACKNOWLEDGED" property is added or changed in the alarm component. If the value of any "ACKNOWLEDGED" property in the alarm changes and is greater than or equal to the trigger time of the alarm, then clients SHOULD dismiss or cancel any "alert" presented to the calendar user.

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

```
acknowledged = "ACKNOWLEDGED" acknowledgedparam ":" datetime CRLF
```

```
acknowledgedparam = *(  
    ; the following is OPTIONAL,  
    ; and MAY occur more than once  
  
    (";" other-param)  
)
```

Example: The following is an example of this property:

```
ACKNOWLEDGED:20090604T084500Z
```

## 6. Snoozing Alarms

Users often want to "snooze" an alarm, and this specification defines a standard approach to accomplish that.

To "snooze" an alarm, clients create a new "VALARM" component within the parent component of the "VALARM" that was triggered and is being "snoozed" (i.e., as a "sibling" component of the "VALARM" being snoozed). The new "VALARM" MUST be set to trigger at the user's chosen "snooze" interval after the original alarm triggered. Clients SHOULD use an absolute "TRIGGER" property with a "DATE-TIME" value specified in UTC.

When the "snooze" alarm is triggered and dismissed the client SHOULD remove the corresponding "VALARM" component, or set the "ACKNOWLEDGED" property (see Section 5.1). Alternatively, if the "snooze" alarm is itself "snoozed", the client SHOULD remove the

original "snooze" alarm and create a new one, with the appropriate trigger time and relationship set.

## 7. Alarm Proximity Trigger

VALARMS are currently triggered when a specific date-time is reached. It is also desirable to be able to trigger alarms based on location, e.g. when arriving at or departing from a particular location.

This specification adds the following properties to "VALARM" components to indicate when an alarm can be triggered based on location.

"PROXIMITY" - indicates that a location based trigger is to be used and which direction of motion is used for the trigger

"STRUCTURED-LOCATION" [I-D.ietf-calext-eventpub-extensions] - used to indicate the actual location to trigger off, specified using a geo: URI [RFC5870] which allows for two or three coordinate values with an optional uncertainty

```
alarmprop      /= *(  
    ; the following is OPTIONAL,  
    ; but MUST NOT occur more than once  
  
    proximity /  
  
    ; the following is OPTIONAL,  
    ; and MAY occur more than once, but only  
    ; when a PROXIMITY property is also present  
  
    structured-location  
  
    )
```

Typically, when a "PROXIMITY" property is used there is no need to specify a time-based trigger using the "TRIGGER" property. However, since "TRIGGER" is defined as a required property for a "VALARM" component, for backwards compatibility it has to be present, but ignored. To indicate a "TRIGGER" that is to be ignored, clients SHOULD use a value a long time in the past. A value of "19760401T005545Z" has been commonly used for this purpose.



### 7.1. Proximity Property

Property Name: PROXIMITY

Purpose: This property indicates that a location based trigger is applied to an alarm.

Value Type: TEXT

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

Conformance: This property can be specified within "VALARM" calendar components.

Description: This property is used to indicate that an alarm has a location-based trigger. Its value identifies the direction of motion used to trigger the alarm. One or more location values are set using "STRUCTURED-LOCATION" properties.

When the property value is set to "ARRIVE", the alarm is triggered when the calendar user agent arrives in the vicinity of any of the specified locations. When set to "DEPART", the alarm is triggered when the calendar user agent departs from the vicinity of any specified locations.

When the property value is set to "CONNECT", the alarm is triggered when the calendar user agent connects to a Bluetooth(R) [BTcore]-enabled automobile. When set to "DISCONNECT", the alarm is triggered when the calendar user agent disconnects from a Bluetooth(R)-enabled automobile.

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

```
proximity = "PROXIMITY" proximityparam ":" proximityvalue CRLF
```

```
proximityparam = *(
```

```
    ; the following is OPTIONAL,  
    ; and MAY occur more than once
```

```
    (";" other-param)
```

```
)
```

```
proximityvalue = "ARRIVE" / "DEPART" /  
                "CONNECT" / "DISCONNECT" / iana-token / x-name
```

Example: The following is an example of this property:

PROXIMITY:ARRIVE

## 7.2. Example

The following example shows a "VALARM" component with a proximity trigger set to trigger when the device running the calendar user agent leaves the vicinity defined by the structured location property. Note use of the "u=" parameter with the "geo" URI to define the precision of the location determination.

```
BEGIN:VALARM
UID:77D80D14-906B-4257-963F-85B1E734DBB6
TRIGGER;VALUE=DATE-TIME:19760401T005545Z
ACTION:DISPLAY
DESCRIPTION:Remember to buy milk
TRIGGER;VALUE=DATE-TIME:19760401T005545Z
PROXIMITY:DEPART
STRUCTURED-LOCATION;VALUE=URI:geo:40.443,-79.945;u=10
END:VALARM
```

## 8. Security Considerations

VALARMS, if not monitored properly, can be used to "spam" users and/or leak personal information. For instance, an unwanted audio or display alert could be considered spam. Or an email alert could be used to leak a user's location to a third party or to send unsolicited email to multiple users. Therefore, CalDAV clients and servers that accept iCalendar data from a third party (e.g. via iTIP [RFC5546], a subscription feed, or a shared calendar) SHOULD remove all VALARMS from the data prior to storing in their calendar system.

## 9. IANA Considerations

### 9.1. Property Registrations

This document defines the following new iCalendar properties to be added to the registry defined in Section 8.2.3 of [RFC5545]:

Property	Status	Reference
ACKNOWLEDGED	Current	RFCXXXX, Section 5.1
PROXIMITY	Current	RFCXXXX, Section 7.1

## 9.2. Proximity Value Registry

This document creates a new iCalendar registry for values of the "PROXIMITY" property:

Value	Status	Reference
ARRIVE	Current	RFCXXXX, Section 7.1
DEPART	Current	RFCXXXX, Section 7.1
CONNECT	Current	RFCXXXX, Section 7.1
DISCONNECT	Current	RFCXXXX, Section 7.1

## 10. Acknowledgments

This specification came about via discussions at the Calendaring and Scheduling Consortium. Also, thanks to the following for providing feedback: Bernard Desruisseaux, Mike Douglass, Jacob Farkas, Jeffrey Harris, and Ciny Joy.

## 11. References

### 11.1. Normative References

- [I-D.ietf-calext-eventpub-extensions]  
Douglass, M., "Event Publishing Extensions to iCalendar",  
draft-ietf-calext-eventpub-extensions-11 (work in  
progress), February 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault,  
"Calendaring Extensions to WebDAV (CalDAV)", RFC 4791,  
DOI 10.17487/RFC4791, March 2007,  
<<https://www.rfc-editor.org/info/rfc4791>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and  
Scheduling Core Object Specification (iCalendar)",  
RFC 5545, DOI 10.17487/RFC5545, September 2009,  
<<https://www.rfc-editor.org/info/rfc5545>>.

- [RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <<https://www.rfc-editor.org/info/rfc5870>>.
- [RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <<https://www.rfc-editor.org/info/rfc7986>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 11.2. Informative References

- [BTcore] Bluetooth Special Interest Group, "Bluetooth Core Specification Version 5.0", December 2016, <<https://www.bluetooth.com/specifications/bluetooth-core-specification>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.

### 11.3. URIs

- [1] <https://tools.ietf.org/html/bcp14>

## Appendix A. Change History (To be removed by RFC Editor before publication)

### Changes in -05:

1. Added Murchison as editor.
2. Updated keywords boilerplate.
3. Added reference to UID security/privacy recommendations.
4. Removed default alarms.
5. Removed ALARM-AGENT property.
6. Added text about using TRIGGER value in the past in addition to ACTION:NONE to have a default alarm be ignored.
7. Removed text about related alarms.

8. Removed URL alarm action.
9. Added reference to draft-ietf-calex-ext-eventpub-extensions for STRUCTURED-LOCATION.
10. Added CONNECT and DISCONNECT PROXIMITY property values.
11. Added Security Considerations.
12. Editorial fixes.

Changes in -04:

1. Changed "ID" to "AGENT-ID".
2. Add more text on using "ACKNOWLEDGED" property.
3. Add "RELATED-TO" as a valid property for VALARMS.
4. Add "SNOOZE" relationship type for use with VALARMS.
5. State that "TRIGGER" is typically ignored in proximity alarms.
6. Added "PROXIMITY" value registry.
7. Added a lot more detail on default alarms including new action and property.

Changes in -03: none - resubmission of -02

Changes in -02:

1. Updated to 5545 reference.
2. Clarified use of absolute trigger in UTC in snooze alarms
3. Snooze alarms should be removed when completed
4. Removed status and replaced last-triggered by acknowledged property
5. Added location-based trigger
6. IANA registry tables added

Changes in -01:

1. Removed DESCRIPTION as an allowed property in the URI alarm.

2. Added statement about what to do when ALARM-AGENT is not present.
3. Allow multiple ALARM-AGENT properties to be present.
4. Removed SNOOZE-UNTIL - snoozing now accomplished by creating a new VALARM.
5. Remove VALARM by reference section.
6. Added more detail to CalDAV default alarms.

#### Authors' Addresses

Cyrus Daboo  
Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
USA

Email: [cyrus@daboo.name](mailto:cyrus@daboo.name)  
URI: <http://www.apple.com/>

Kenneth Murchison (editor)  
FastMail US LLC  
1429 Walnut St, Suite 1201  
Philadelphia, PA 19102  
USA

Email: [murch@fastmailteam.com](mailto:murch@fastmailteam.com)  
URI: <http://www.fastmail.com/>

calext  
Internet-Draft  
Updates: 6638 (if approved)  
Intended status: Standards Track  
Expires: September 9, 2019

B. Gondwana, Ed.  
FastMail  
March 8, 2019

CalDAV Extension for scheduling controls  
draft-gondwana-caldav-scheduling-controls-01

Abstract

This document adds headers to control and restrict the scheduling behaviour of CalDAV servers when updating calendaring resources.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 9, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Conventions Used In This Document . . . . .	3
3. Extending the CalDAV OPTIONS response . . . . .	3
3.1. Example: Using OPTIONS for the Discovery of Scheduling Controls Support . . . . .	3
4. New headers . . . . .	3
4.1. Scheduling header . . . . .	3
4.2. Schedule-User-Address header . . . . .	4
5. Implementation considerations . . . . .	5
6. IANA Considerations . . . . .	5
7. Security Considerations . . . . .	5
8. Acknowledgments . . . . .	5
9. Version History . . . . .	5
9.1. v01, 2019-03-08 . . . . .	5
9.2. v00, 2019-02-08 . . . . .	5
10. Normative References . . . . .	6
Author's Address . . . . .	6

## 1. Introduction

[RFC6638] defines automatic scheduling operations for resources stored on [!@RFC4791] CalDAV servers.

[RFC6638] defines the "Schedule-Reply" header in Section 8.1, however this header is not sufficient for controlling scheduling in all cases.

Cases where it might be necessary to update the data store on a server without causing scheduling messages to be sent include backup after a data loss event on the server, or importing calendar events from another system.

Calendar server operators deal with these other needs by either using a different method than CalDAV to update their server, or by adding a custom method to suppress scheduling. This document defines a standard method to suppress scheduling, allowing CalDAV to be directly used for restores and imports.

Complex sites can have users who have multiple aliases, and in the most complex cases, a user may have multiple identities who are present on a scheduling event as organizer and/or attendee. When an event is updated over CalDAV, the server must calculate or guess which of those addresses the current user is acting as. This document defines a header which allows the client to inform the server precisely which address they are acting as when adding, modifying or removing a resource.



## 2. Conventions Used In This Document

In examples, "C:" indicates data sent by a client that is connected to a server. "S:" indicates data sent by the server to the client.

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] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

## 3. Extending the CalDAV OPTIONS response

A server supporting the features described in this document MUST include "scheduling-controls" as a field in the DAV response header from an OPTIONS request. A value of "scheduling-controls" in the DAV response header indicates to clients that the server supports all the requirements specified in this document.

### 3.1. Example: Using OPTIONS for the Discovery of Scheduling Controls Support

Request:

```
OPTIONS /home/brong/calendars/ HTTP/1.1
Host: cal.example.com
```

Response:

```
HTTP/1.1 200 OK
Allow: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, COPY, MOVE
Allow: PROPFIND, PROPPATCH, LOCK, UNLOCK, REPORT, ACL
DAV: 1, 2, 3, access-control, calendar-access,
    scheduling-controls
Date: Thu, 8 Feb 2019 10:16:37 GMT
Content-Length: 0
```

## 4. New headers

This document adds two new headers for use on PUT, PROPPATCH and DELETE:

### 4.1. Scheduling header

Scheduling: {all|none|internal-only|external-only|X-...}

Default: all

Not providing this header, or providing the value of "all", instructs the server to follow the behaviour in [RFC6638] Section 3.2.

Providing the value "none" instructs the server to perform no scheduling at all, and to just store the event (useful for restoring from backup)

The value "internal-only" instructs the server to update the events in other calendars within its system where that can be done silently, but not to send visible notifications to users (where permitted by policy). This is useful when importing multiple related calendars into a new system without flooding external parties with notifications.

The value "external-only" instructs the server to import the data without updating local calendars, but to send notifications to external attendees so they are aware of the event. This is useful when migrating calendar events to a new system where external parties need to have a way to update their participation status in the new system.

e.g.

Scheduling: none

TODO: specify error codes

#### 4.2. Schedule-User-Address header

Schedule-User-Address: URI

Default: not present

If this header is not present, the server will calculate the address from the authenticated user, or from the CALDAV:schedule-user-address property on the calendar or principal.

If this header is provided, it overrides the server's internal calculation, and informs the server to perform any scheduling as the specified user.

TODO: specify error codes

e.g.

Schedule-User-Address: mailto:foo@example.com

## 5. Implementation considerations

Any server implementing this extension MUST ensure it has a way to validate Schedule-User-Address settings.

## 6. IANA Considerations

TODO: IANA request for OPTIONS item

TODO: IANA request for named headers

## 7. Security Considerations

The "Scheduling" header only allows reduction of the cases in which the server will create scheduling requests. This is generally good for user privacy, allowing copies of events to be updated without notifying the owner or attendees. This is particularly valuable for cleaning up spam.

The "Schedule-User-Address" header allows the client to override the server choice of address for the user to act as. Servers MUST ensure that the authenticated user has permission to act as the specified address, as well as applying any local policy limitations.

## 8. Acknowledgments

- o Lucia Kristiansen, Google
- o CalConnect
- o The calext working group

## 9. Version History

Remove before publishing

### 9.1. v01, 2019-03-08

- o correct name in acknowledgements

### 9.2. v00, 2019-02-08

- o Initial draft based on discussion at CalConnect about Google and FastMail private implementations.

## 10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6638] Daboo, C. and B. Desruisseaux, "Scheduling Extensions to CalDAV", RFC 6638, DOI 10.17487/RFC6638, June 2012, <<https://www.rfc-editor.org/info/rfc6638>>.

## Author's Address

Bron Gondwana (editor)  
FastMail  
Level 2, 114 William St  
Melbourne VIC 3000  
Australia

Email: [brong@fastmailteam.com](mailto:brong@fastmailteam.com)  
URI: <https://www.fastmail.com>

Calendering Extensions  
Internet-Draft  
Intended status: Informational  
Expires: September 6, 2019

C. Daboo  
Apple  
A. Quillaud  
Oracle  
K. Murchison, Ed.  
FastMail  
March 5, 2019

CalDAV Managed Attachments  
draft-ietf-calext-caldav-attachments-04

Abstract

This specification defines an extension to the calendar access protocol (CalDAV) to allow attachments associated with iCalendar data to be stored and managed on the server.

This specification documents existing code deployed by multiple vendors. It is published as an Informational specification rather than Standards Track due to its noncompliance with multiple best current practices of HTTP.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Rationale for Informational Status . . . . .	4
2. Conventions Used in This Document . . . . .	4
3. Overview . . . . .	4
3.1. Requirements . . . . .	5
3.2. Discovering Support for Managed Attachments . . . . .	5
3.3. POST Request for Managing Attachments . . . . .	5
3.3.1. action= Query Parameter . . . . .	6
3.3.2. rid= Query Parameter . . . . .	6
3.3.3. managed-id= Query Parameter . . . . .	7
3.4. Adding attachments . . . . .	7
3.5. Updating Attachments . . . . .	10
3.6. Removing Attachments via POST . . . . .	13
3.7. Adding Existing Managed Attachments via PUT . . . . .	15
3.8. Updating Attachments via PUT . . . . .	15
3.9. Removing Attachments via PUT . . . . .	15
3.10. Retrieving Attachments . . . . .	16
3.11. Error Handling . . . . .	16
3.12. Additional Considerations . . . . .	17
3.12.1. Quotas . . . . .	17
3.12.2. Access Control . . . . .	17
3.12.3. Redirects . . . . .	18
3.12.4. Processing Time . . . . .	18
3.12.5. Automatic Clean-Up by Servers . . . . .	18
3.12.6. Sending Scheduling Messages with Attachments . . . . .	18
3.12.7. Migrating Calendar Data . . . . .	18
4. Modifications to iCalendar Syntax . . . . .	19
4.1. SIZE Property Parameter . . . . .	19
4.2. FILENAME Property Parameter . . . . .	19
4.3. MANAGED-ID Property Parameter . . . . .	20
5. Additional Message Header Fields . . . . .	20
5.1. Cal-Managed-ID Response Header Field . . . . .	20
6. Additional WebDAV Properties . . . . .	21
6.1. CALDAV:managed-attachments-server-URL property . . . . .	21
6.2. CALDAV:max-attachment-size property . . . . .	22
6.3. CALDAV:max-attachments-per-resource property . . . . .	23
7. Implementation Status . . . . .	23
8. Security Considerations . . . . .	25
9. IANA Considerations . . . . .	26

9.1. Parameter Registrations . . . . .	26
9.2. Message Header Field Registrations . . . . .	26
9.2.1. Cal-Managed-ID . . . . .	26
10. Acknowledgments . . . . .	27
11. References . . . . .	27
11.1. Normative References . . . . .	27
11.2. Informative References . . . . .	28
11.3. URIs . . . . .	29
Appendix A. Change History (To be removed by RFC Editor before publication) . . . . .	29
Appendix B. Example Involving Recurring Events . . . . .	32
Authors' Addresses . . . . .	39

## 1. Introduction

The iCalendar [RFC5545] data format is used to represent calendar data and is used with iCalendar Transport-independent Interoperability Protocol (iTIP) [RFC5546] to handle scheduling operations between calendar users.

[RFC4791] defines the Calendaring Extensions to WebDAV (CalDAV), based on HTTP [RFC7230], for accessing calendar data stored on a server.

Calendar users often want to include attachments with their calendar data events or tasks (for example a copy of a presentation, or the meeting agenda). iCalendar provides an "ATTACH" property whose value is either the inline Base64 encoded attachment data, or a URL specifying the location of the attachment data.

Use of inline attachment data is not ideal with CalDAV because the data would need to be uploaded to the server each time a change to the calendar data is done - even minor changes such as a change to the summary. Whilst a client could choose to use a URL value instead, the problem then becomes where and how the client discovers an appropriate URL to use and how it ensures that only those attendees listed in the event or task are able to access it.

This specification solves this problem by having the client send the attachment to the server, separately from the iCalendar data, and the server takes care of adding appropriate "ATTACH" properties in the iCalendar data as well as managing access privileges. The server can also provide additional information to the client about each attachment in the iCalendar data, such as the size and an identifier.

### 1.1. Rationale for Informational Status

Although this extension to CalDAV has wide deployment, its design does not comply with some of the best current practices of HTTP, namely:

- o All operations on attachments are modeled as HTTP POST operations, where the actual type of operation is specified using a query parameter, instead of using separate HTTP POST, PUT, and DELETE methods where appropriate.
- o Specific query strings are hardwired into the protocol in violation of Section 2.4 of [RFC7320].

Additionally, this extension misuses the Content-Disposition header field [RFC6266] as a request header field to convey a filename for an attachment rather than using an existing request header field suitable for that purpose, such as "Slug" (see Section 9.7 of [RFC5023]).

Rather than creating interoperability problems with deployed code by updating the design of this extension to be compliant with best current practices and to allow this specification to be placed on the Standards Track, it was decided to simply document how existing implementations interoperate and to publish the document as Informational.

### 2. 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 [1] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

### 3. Overview

There are four main operations a client needs to do with attachments for calendar data: add, update, remove, and retrieve. The first three operations are carried out by the client issuing an HTTP POST request on the calendar object resource to which the attachment is associated and specifying the appropriate "action" query parameter (see Section 3.3). In the case of the remove operation, the client



can alternatively directly update the calendar object resource and remove the relevant "ATTACH" properties (see Section 3.9). The retrieve operation is accomplished by simply issuing an HTTP GET request targeting the attachment URI specified by the calendar resource's "ATTACH" property (see Section 3.10).

iCalendar data stored in a CalDAV calendar object resource can contain multiple components when recurrences are involved. In such a situation, the client needs to be able to target a specific recurrence instance or multiple instances when adding or deleting attachments. As a result, the POST request needs to provide a way for the client to specify which recurrence instances should be targeted for the attachment operation. This is accomplished through use of additional query parameters on the POST request-URI.

### 3.1. Requirements

A server that supports the features described in this specification is REQUIRED to support the CalDAV "calendar-access" [RFC4791] features.

In addition, such a server SHOULD support the "return=representation" Prefer header field [RFC7240] preference on successful HTTP PUT and POST requests targeting existing calendar object resources, by returning the new representation of that calendar resource (including its new ETag header field value) in the response.

### 3.2. Discovering Support for Managed Attachments

A server supporting the features described in this specification MUST include "calendar-managed-attachments" as a token in the DAV response header field (as defined in Section 10.1 of [RFC4918]) from an OPTIONS request on a calendar home collection.

A server might choose to not support storing managed attachments on a per-recurrence instance basis (i.e., they can only be added to all instances as a whole). If that is the case, the server MUST also include "calendar-managed-attachments-no-recurrence" as a token in the DAV response header field from an OPTIONS request on a calendar home collection. When that field is present, clients MUST NOT attempt any managed attachment operations that target specific recurrence instances.

### 3.3. POST Request for Managing Attachments

An HTTP POST request is used to add, update, or remove attachments. These requests are subject to the preconditions listed in

Section 3.11. The request-URI will contain various query parameters to specify the behavior.

#### 3.3.1. action= Query Parameter

The "action" query parameter is used to identify which attachment operation the client is requesting. This parameter **MUST** be present once on each POST request used to manage attachments. One of these three values **MUST** be used:

attachment-add Indicates an operation that is adding an attachment to a calendar object resource. See Section 3.4 for more details.

attachment-update Indicates an operation that is updating an existing attachment on a calendar object resource. See Section 3.5 for more details.

attachment-remove Indicates an operation that is removing an attachment from a calendar object resource. See Section 3.6 for more details.

Example:

`https://calendar.example.com/events/1.ics?action=attachment-add`

#### 3.3.2. rid= Query Parameter

The "rid" query parameter is used to identify which recurrence instances are being targeted by the client for the attachment operation. This query parameter **MUST** contain one or more items, separated by commas (0x2C). The item values can be in one of two forms:

Master instance The value "M" (case-insensitive) refers to the "master" recurrence instance, i.e., the component that does not include a "RECURRENCE-ID" property. This item **MUST** be present only once.

Specific instance A specific iCalendar instance is targeted by using its "RECURRENCE-ID" value as the item value. That value **MUST** correspond to the RECURRENCE-ID value as stored in the calendar object resource (i.e. without any conversion to UTC). If multiple items of this form are used, they **MUST** be unique values. For example, to target a recurrence defined by property RECURRENCE-ID;TZID=America/Montreal:20111022T160000, the query parameter `rid=20111022T160000` would be used.

If the "rid" query parameter is not present, all recurrence instances in the calendar object resource are targeted.

The "rid" query parameter MUST NOT be present in the case of an update operation, or if the server chooses not to support per-recurrence instance managed attachments (see Section 3.1).

Example (targeting the master instance and a specific overridden instance):

```
https://calendar.example.com/events/1.ics?  
action=attachment-add&rid=M,20111022T160000
```

### 3.3.3. managed-id= Query Parameter

The "managed-id" query parameter is used to identify which "ATTACH" property is being updated or removed. The value of this query parameter MUST match the "MANAGED-ID" (Section 4.3) property parameter value on the "ATTACH" property in the calendar object resource instance(s) targeted by the request.

The "managed-id" query parameter MUST NOT be present in the case of an add operation.

Example:

```
https://calendar.example.com/events/1.ics?  
action=attachment-update&managed-id=aUNhbGVuZGFy
```

### 3.4. Adding attachments

To add an attachment to an existing calendar object resource, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.
  - A. The request-URI will include an "action" query parameter with the value "attachment-add" (see Section 3.3.1).
  - B. If all recurrence instances are having an attachment added, the "rid" query parameter is not present in the request-URI. If one or more specific recurrence instances are targeted, then the request-URI will include a "rid" query parameter containing the list of instances (see Section 3.3.2).
  - C. The body of the request contains the data for the attachment.

- D. The client **MUST** include a valid Content-Type header field describing the media type of the attachment (as required by HTTP).
  - E. The client **SHOULD** include a Content-Disposition header field [RFC6266] with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment. Note that the use of Content-Disposition as a request header field is nonstandard and specific to this protocol.
  - F. The client **MAY** include a Prefer header field [RFC7240] with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.
2. When the server receives the POST request it does the following:
- A. Validates that any recurrence instances referred to via the "rid" query parameter are valid for the calendar object resource being targeted.
  - B. Stores the supplied attachment data into a resource and generates an appropriate URI for clients to access the resource.
  - C. For each affected recurrence instance in the calendar object resource targeted by the request, the server adds an "ATTACH" property, whose value is the URI of the stored attachment. The "ATTACH" property **MUST** contain a "MANAGED-ID" parameter whose value is a unique identifier (within the context of the server as a whole). The "ATTACH" property **SHOULD** contain an "FMCTYPE" parameter whose value matches the Content-Type header field value from the request. The "ATTACH" property **SHOULD** contain a "FILENAME" parameter whose value matches the Content-Disposition header field "filename" parameter value from the request, taking into account the restrictions expressed in Section 4.2. The "ATTACH" property **SHOULD** include a "SIZE" parameter whose value represents the size in octets of the attachment. If a specified recurrence instance does not have a matching component in the calendar object resource, then the server **MUST** modify the calendar object resource to include an overridden component with the appropriate "RECURRENCE-ID" property.
  - D. Upon successful creation of the attachment resource, and modification of the targeted calendar object resource, the

server MUST return an appropriate HTTP success status response and include a "Cal-Managed-ID" header field containing the "MANAGED-ID" parameter value of the newly created "ATTACH" property. The client can use the "Cal-Managed-ID" header field value to correlate the attachment with "ATTACH" properties added to the calendar object resource. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

In the following example, the client adds a new attachment to a non recurring event and asks the server (via the Prefer [RFC7240] header field) to return the modified version of that event in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment; filename=agenda.html
Content-Length: xxxx
Prefer: return=representation
```

```
<html>
  <body>
    <h1>Agenda</h1>
  </body>
</html>
```

>> Response <<

```
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://cal.example.com/events/64.ics
ETag: "123456789-000-111"
Cal-Managed-ID: 97S

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
DTSTAMP:20120201T203412Z
DTSTART:20120714T170000Z
DTEND:20120715T040000Z
SUMMARY:One-off meeting
ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
  FILENAME=agenda.html:https://cal.example.com/attach/64/34X22R
END:VEVENT
END:VCALENDAR
```

### 3.5. Updating Attachments

When an attachment is updated the server **MUST** change the associated "MANAGED-ID" parameter and **MAY** change the "ATTACH" property value. With this approach, clients are able to determine when an attachment has been updated by some other client by looking for a change to either the "ATTACH" property value, or the "MANAGED-ID" parameter value.

To change the data of an existing managed attachment in a calendar object resource, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.
  - A. The request-URI will include an "action" query parameter with the value "attachment-update" (see Section 3.3.1).
  - B. The request-URI will include a "managed-id" query parameter with the value matching that of the "MANAGED-ID" parameter for the "ATTACH" property being updated (see Section 3.3.3).
  - C. The body of the request contains the updated data for the attachment.

- D. The client **MUST** include a valid Content-Type header field describing the media type of the attachment (as required by HTTP).
  - E. The client **SHOULD** include a Content-Disposition header field [RFC6266] with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment.
  - F. The client **MAY** include a Prefer header field [RFC7240] with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.
2. When the server receives the POST request it does the following:
- A. Validates that the "managed-id" query parameter is valid for the calendar object resource.
  - B. Updates the content of the attachment resource corresponding to that managed-id with the supplied attachment data.
  - C. For each affected recurrence instance in the calendar object resource targeted by the request, the server updates the "ATTACH" property whose "MANAGED-ID" property parameter value matches the "managed-id" query parameter. The "MANAGED-ID" parameter value is changed to allow other clients to detect the update, and the property value (attachment URI) might also be changed. The "ATTACH" property **SHOULD** contain a "FMCTYPE" parameter whose value matches the Content-Type header field value from the request - this could differ from the original value if the media type of the updated attachment is different. The "ATTACH" property **SHOULD** contain a "FILENAME" parameter whose value matches the Content-Disposition header field "filename" parameter value from the request, taking into account the restrictions expressed in Section 4.2. The "ATTACH" property **SHOULD** include a "SIZE" parameter whose value represents the size in octets of the updated attachment.
  - D. Upon successful update of the attachment resource, and modification of the targeted calendar object resource, the server **MUST** return an appropriate HTTP success status response, and include a "Cal-Managed-ID" header field containing the new value of the "MANAGED-ID" parameter. The client can use the "Cal-Managed-ID" header field value to correlate the attachment with "ATTACH" properties added to

the calendar object resource. If the client included a `Prefer` header field with the `"return=representation"` preference in the request, the server **SHOULD** return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent `GET` request to refresh any local cache.

The update operation does not take a `"rid"` parameter and does not add, or remove, any `"ATTACH"` property in the targeted calendar object resource. To link an existing attachment to a new instance, the client simply does a `PUT` on the calendar object resource, adding an `"ATTACH"` property which duplicates the existing one (see Section 3.7).

In the following example, the client updates an existing attachment and asks the server (via the `Prefer [RFC7240]` header field) to return the updated version of that event in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-update&managed-id=97S HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment; filename=agenda.html
Content-Length: xxxx
Prefer: return=representation
```

```
<html>
  <body>
    <h1>Agenda</h1>
    <p>Discuss attachment draft</p>
  </body>
</html>
```



>> Response <<

```
HTTP/1.1 200 OK
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyz
Content-Location: https://cal.example.com/events/64.ics
Cal-Managed-ID: 98S
ETag: "123456789-000-222"

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
DTSTAMP:20120201T203412Z
DTSTART:20120714T170000Z
DTEND:20120715T040000Z
SUMMARY:One-off meeting
ATTACH;MANAGED-ID=98S;FMTTYPE=text/html;SIZE=xxxy;
  FILENAME=agenda.html:https://cal.example.com/attach/64/34X22R
END:VEVENT
END:VCALENDAR
```

### 3.6. Removing Attachments via POST

To remove an existing attachment from a calendar object, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.
  - A. The request-URI will include an "action" query parameter with the value "attachment-remove" (see Section 3.3.1).
  - B. If all recurrence instances are having an attachment removed, the "rid" query parameter is not present in the request-URI. If one or more specific recurrence instances are targeted, then the request-URI will include a "rid" query parameter containing the list of instances (see Section 3.3.2).
  - C. The request-URI will include a "managed-id" query parameter with the value matching that of the "MANAGED-ID" property parameter for the "ATTACH" property being removed (see Section 3.3.3).
  - D. The body of the request will be empty.

- E. The client MAY include a Prefer header field [RFC7240] with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.
2. When the server receives the POST request it does the following:
- A. Validates that any recurrence instances referred to via the "rid" query parameter are valid for the calendar object resource being targeted.
  - B. Validates that the "managed-id" query parameter is valid for the calendar object resource and specific instances being targeted.
  - C. For each affected recurrence instance in the calendar object resource targeted by the request, the server removes the matching "ATTACH" property. Note that if a specified recurrence instance does not have a matching component in the calendar object resource, then the server MUST modify the calendar object resource to include an overridden component with the appropriate "RECURRENCE-ID" property, and the matching "ATTACH" property removed. This later case is actually valid only if the master component does include the referenced "ATTACH" property.
  - D. If the attachment resource is no longer referenced by any instance of the calendar object resource, the server can delete the attachment resource to free up storage space.
  - E. Upon successful removal of the attachment resource and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

In the following example, the client deletes an existing attachment by passing its managed-id in the request. The Prefer [RFC7240] header field is not set in the request so the calendar object resource data is not returned in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-remove&managed-id=98S HTTP/1.1
Host: cal.example.com
Content-Length: 0
```

>> Response <<

```
HTTP/1.1 204 No Content
Content-Length: 0
```

### 3.7. Adding Existing Managed Attachments via PUT

Clients can make use of existing managed attachments by adding the corresponding "ATTACH" property to calendar object resources (subject to the restrictions described in Section 3.12.2)

If a managed attachment is used in more than calendar resource, servers SHOULD NOT change either the "MANAGED-ID" parameter value or the "ATTACH" property value for these attachments - this ensures that clients do not have to download the attachment data again if they already have it cached. Additionally, servers SHOULD validate "SIZE" parameter values and replace incorrect values with the actual sizes of existing attachments.

These PUT requests are subject to the preconditions listed in Section 3.11.

### 3.8. Updating Attachments via PUT

Servers MUST NOT allow clients to update attachment data directly via a PUT on the attachment URI (or via any other HTTP method that modifies content). Instead, attachments can only be updated via use of POST requests on the calendar data.

### 3.9. Removing Attachments via PUT

Clients can remove attachments by simply re-writing the calendar object resource data to remove the appropriate "ATTACH" properties. Servers MUST NOT allow clients to delete attachments directly via a DELETE request on the attachment URI.

### 3.10. Retrieving Attachments

Clients retrieve attachments by issuing an HTTP GET request using the value of the corresponding "ATTACH" property as the request-URI, taking into account the substitution mechanism associated with the "CALDAV:managed-attachments-server-URL" property (see Section 6.1).

### 3.11. Error Handling

This specification creates additional preconditions for the POST method.

The new preconditions are:

(CALDAV:max-attachment-size): The attachment submitted in the POST request MUST have an octet size less than or equal to the value of the CALDAV:max-attachment-size property value (Section 6.2) on the calendar collection of the target calendar resource;

(CALDAV:max-attachments-per-resource): The addition of the attachment submitted in the POST request MUST result in the target calendar resource having a number of managed attachments less than or equal to the value of the CALDAV:max-attachments-per-resource property value (Section 6.3) on the calendar collection of the target calendar resource;

(CALDAV:valid-action): The action query parameter in the POST request MUST contain one of "attachment-add", "attachment-update", or "attachment-remove".

(CALDAV:valid-rid): The rid query parameter in the POST request MUST NOT be present for an attachment-update action, and MUST contain the value "M" and/or values corresponding to "RECURRENCE-ID" property values in the iCalendar data targeted by the request.

(CALDAV:valid-managed-id): The managed-id query parameter in the POST request MUST NOT be present for an attachment-add action, and MUST contain a value corresponding to a "MANAGED-ID" property parameter value in the iCalendar data targeted by the request.

A POST request to add, modify, or delete a managed attachment results in an implicit modification of the targeted calendar resource (equivalent of a PUT). As a consequence, clients should also be prepared to handle preconditions associated with this implicit PUT. This includes (but is not limited to):

(CALDAV:max-resource-size) (from Section 5.3.2.1 of [RFC4791])

(DAV:quota-not-exceeded) (from Section 6 of [RFC4331])

(DAV:sufficient-disk-space) (from Section 6 of [RFC4331])

A PUT request to add or modify an existing calendar object resource can make reference to an existing managed attachment. The following new preconditions are defined:

(CALDAV:valid-managed-id-parameter): a "MANAGED-ID" property parameter value in the iCalendar data in the PUT request is not valid (e.g., does not match any existing managed attachment).

If a precondition for a request is not satisfied:

1. The response status of the request MUST either be 403 (Forbidden), if the request should not be repeated because it will always fail, or 409 (Conflict), if it is expected that the user might be able to resolve the conflict and resubmit the request.
2. The appropriate XML element MUST be returned as the child of a top-level DAV:error element in the response body.

### 3.12. Additional Considerations

#### 3.12.1. Quotas

The WebDAV Quotas [RFC4331] specification defines two live WebDAV properties (DAV:quota-available-bytes and DAV:quota-used-bytes) to communicate storage quota information to clients. Server implementations MAY choose to include managed attachments sizes when calculating the amount of storage used by a particular resource.

#### 3.12.2. Access Control

Access to the managed attachments store in a calendar object resource SHOULD be restricted to only those calendar users who have access to that calendar object either directly, or indirectly (via being an attendee who would receive a scheduling message).

When accessing a managed attachment, clients SHOULD be prepared to authenticate with the server storing the attachment resource. The credentials required to access the managed attachment store could be different from the ones used to access the CalDAV server.

This specification only allows organizers of scheduled events to add managed attachments. Servers MUST prevent attendees of scheduled events from adding, updating or removing managed attachments. In

addition, the server **MUST** prevent a calendar user from re-using a managed attachment (based on its managed-id value), unless that user is the one who originally created the managed attachment.

#### 3.12.3. Redirects

For POST requests that add or update attachment data, the server **MAY** issue a 307 (Temporary Redirect) [RFC7231] or 308 (Permanent Redirect) [RFC7538] response to require the client to re-issue the POST request using a different request-URI. As a result, clients **SHOULD** use the "100-continue" expectation defined in Section 5.1.1 of [RFC7231]. Using this mechanism ensures that, if a redirect does occur, the client does not needlessly send the attachment data.

#### 3.12.4. Processing Time

Clients can expect servers to take a while to respond to POST requests that include large attachment bodies. Servers **SHOULD** use the "102 (Processing)" interim response defined in Section 10.1 of [RFC2518] to keep the client connection alive if the POST request will take significant time to complete.

#### 3.12.5. Automatic Clean-Up by Servers

Servers **MAY** automatically remove attachment data, for example to regain the storage taken by unused attachments, or as the result of a virus scanning. When doing so they **SHOULD NOT** modify calendar data referencing those attachments. Instead they **SHOULD** respond with "410 (Gone)" to any request on the removed attachment URI.

#### 3.12.6. Sending Scheduling Messages with Attachments

When a managed attachment is added, updated or removed from a calendar object resource, the server **MUST** ensure that a scheduling message is sent to update any attendees with the changes, as per [RFC6638].

#### 3.12.7. Migrating Calendar Data

When exporting calendar data from a CalDAV server supporting managed attachments, clients **SHOULD** remove all "MANAGED-ID" property parameters from "ATTACH" properties in the calendar data. Similarly when importing calendar data from another source, clients **SHOULD** remove any "MANAGED-ID" property parameters on "ATTACH" properties (failure to do so will likely result in the server removing those properties automatically).

## 4. Modifications to iCalendar Syntax

### 4.1. SIZE Property Parameter

Parameter Name: SIZE

Purpose: To specify the size of an attachment.

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

```
sizeparam = "SIZE" "=" paramtext  
; positive integers
```

Description: This property parameter MAY be specified on "ATTACH" properties. It indicates the size in octets of the corresponding attachment data. Since iCalendar integer values are restricted to a maximum value of 2147483647, the current parameter is defined as text to allow an extended range to be used.

Example:

```
ATTACH;SIZE=1234:https://attachments.example.com/abcd.txt
```

### 4.2. FILENAME Property Parameter

Parameter Name: FILENAME

Purpose: To specify the file name of a managed attachment.

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

```
filenameparam = "FILENAME" "=" paramtext
```

Description: This property parameter MAY be specified on "ATTACH" properties corresponding to managed attachments. Its value provides information on how to construct a filename for storing the attachment data. This parameter is very similar in nature to the Content-Disposition HTTP header field "filename" parameter and exposes the same security risks. As a consequence, clients MUST follow the guidelines expressed in Section 4.3 of [RFC6266] when consuming this parameter value. Similarly, servers MUST follow those same guidelines before storing a value.

Example:

ATTACH;FILENAME=agenda.html:https://attachments.example.com/rt452S

#### 4.3. MANAGED-ID Property Parameter

Parameter Name: MANAGED-ID

Purpose: To uniquely identify a managed attachment.

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

managedidparam = "MANAGED-ID" "=" paramtext

Description: This property parameter MUST be specified on "ATTACH" properties corresponding to managed attachments. Its value is generated by the server and uniquely identifies a managed attachment within the scope of the CalDAV server. This property parameter MUST NOT be present in the case of non-managed attachments.

Example:

ATTACH;MANAGED-ID=aUNhbGVuZGFy:https://attachments.example.com/abcd.txt

### 5. Additional Message Header Fields

#### 5.1. Cal-Managed-ID Response Header Field

The Cal-Managed-ID response header field provides the value of the MANAGED-ID parameter corresponding to a newly added ATTACH property.

ABNF:

Cal-Managed-ID = "Cal-Managed-ID" ":" paramtext  
; "paramtext" is defined in Section 3.1 of [RFC5545]

Example:

Cal-Managed-ID:aUNhbGVuZGFy

The Cal-Managed-ID header field MUST only be sent by an origin server in response to a successful POST request with an action set to attachment-add or attachment-update. It MUST only appear once in a response and MUST NOT appear in trailers.



The Cal-Managed-ID header field is end to end and MUST be forwarded by intermediaries. Intermediaries MUST NOT insert, delete, or modify a Cal-Managed-ID header field.

## 6. Additional WebDAV Properties

### 6.1. CALDAV:managed-attachments-server-URL property

Name: managed-attachments-server-URL

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

Purpose: Specifies the server base URI to use when retrieving managed attachments.

Protected: This property MUST be protected as only the server can update the value.

COPY/MOVE behavior: This property is only defined on a calendar home collection which cannot be moved or copied.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: This property MAY be defined on a calendar home collection. If present, it contains zero or one DAV:href XML elements.

When one DAV:href element is present, its value MUST be an absolute HTTP URI containing only the scheme (i.e. "https") and authority (i.e. host and port) parts. Whenever a managed attachment is to be retrieved via an HTTP GET, the client MUST construct the actual URL of the attachment by substituting the scheme and authority parts of the attachment URI (as stored in the iCalendar "ATTACH" property) with the present WebDAV property value.

When no DAV:href element is present, the client MUST substitute the scheme and authority parts of the attachment URI with the scheme and authority part of the calendar home collection absolute URI.

In the absence of this property, the client can consider the attachment URI as its actual URL.

Definition:

```
<!ELEMENT managed-attachments-server-URL (DAV:href?)>
```

Example:

```
<C:managed-attachments-server-URL xmlns:D="DAV:"
  xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:href>https://attachstore.example.com</D:href>
</C:managed-attachments-server-URL>
```

## 6.2. CALDAV:max-attachment-size property

Name: max-attachment-size

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

Purpose: Provides a numeric value indicating the maximum attachment size, in octets, that the server is willing to accept when a managed attachment is stored on the server.

Protected: MUST be protected as it indicates limits provided by the server.

COPY/MOVE behavior: This property value MUST be preserved in COPY and MOVE operations.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: The CALDAV:max-attachment-size property is used to specify a numeric value that represents the maximum attachment size, in octets, that the server is willing to accept when a managed attachment is stored on the server. The property is defined on the parent collection of the calendar object resource to which the attachment is associated. Any attempt to store a managed attachment exceeding this size MUST result in an error, with the CALDAV:max-attachment-size precondition (Section 3.11) being violated. In the absence of this property, the client can assume that the server will allow storing an attachment of any reasonable size.

Definition:

```
<!ELEMENT max-attachment-size (#PCDATA)>
<!-- PCDATA value: a numeric value (positive decimal integer) -->
```

Example:

```
<C:max-attachment-size xmlns:C="urn:ietf:params:xml:ns:caldav"
  >102400000</C:max-attachment-size>
```

### 6.3. CALDAV:max-attachments-per-resource property

Name: max-attachments-per-resource

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

Purpose: Provides a numeric value indicating the maximum number of managed attachments across all instances of a calendar object resource stored in a calendar collection.

Protected: MUST be protected as it indicates limits provided by the server.

COPY/MOVE behavior: This property value MUST be preserved in COPY and MOVE operations.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: The CALDAV:max-attachments-per-resource property is used to specify a numeric value that represents the maximum number of managed attachments across all instances of a calendar object resource stored in a calendar collection. Non-managed attachments are not counted toward that limit. The property is defined on the parent collection of the calendar object resource to which the attachment is associated. Any attempt to add a managed attachment that would cause the calendar resource to exceed this limit MUST result in an error, with the CALDAV:max-attachments-per-resource precondition (Section 3.11) being violated. In the absence of this property, the client can assume that the server can handle any number of managed attachments per calendar resource.

Definition:

```
<!ELEMENT max-attachments-per-resource (#PCDATA)>
<!-- PCDATA value: a numeric value (positive decimal integer) -->
```

Example:

```
<C:max-attachments-per-resource
  xmlns:C="urn:ietf:params:xml:ns:caldav"
  >12</C:max-attachments-per-resource>
```

## 7. Implementation Status

< RFC Editor: before publication please remove this section, the reference to [RFC7942], and any resulting "URIs" references subsection. >

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".

#### 7.1. Calendar and Contacts Server

The open source Calendar and Contacts Server [2] project is a standards-compliant server implementing the CalDAV protocol. This production level implementation supports all of the requirements described in this document and successfully interoperates with the Apple Calendar, BusyCal, 2Do, and CalDAVTester client implementations described below. This implementation is freely distributable under the terms of the Apache License, Version 2.0 [3].

#### 7.2. Cyrus Server

The open source Cyrus Server [4] project is a highly scalable enterprise mail system which also supports calendaring. This production level CalDAV implementation supports all of the requirements described in this document and successfully interoperates with the Apple Calendar and CalDAVTester client implementations described below. This implementation is freely distributable under a BSD style license from Computing Services at Carnegie Mellon University [5].

#### 7.3. Oracle Communications Calendar Server

The Oracle Communications Calendar Server [6] project is a standards-compliant, scalable, enterprise-ready calendaring solution. This production level CalDAV implementation supports all of the requirements described in this document and successfully interoperates with the Apple Calendar and CalDAVTester client

implementations described below. This implementation is proprietary and available for a free trial and/or purchase from the vendor.

#### 7.4. Apple Calendar

The widely used Apple Calendar [7] client is a standards-compliant client implementing the CalDAV protocol. This production level implementation supports all the requirements described in this document and successfully interoperates with the Calendar and Contacts Server, Cyrus Server, and Oracle Communications Calendar Server implementations described above. This client implementation is proprietary and is distributed as part of Apple's desktop operating systems.

#### 7.5. BusyCal

BusyCal [8] is a standards-compliant calendar client for MacOS implementing the CalDAV protocol. This implementation supports all of the requirements described in this document and successfully interoperates with the Calendar and Contacts Server and Cyrus Server implementations described above. This implementation is proprietary and available for a free trial and/or purchase from the vendor.

#### 7.6. CalDAVTester

CalDAVTester [9] is an open source test and performance application designed to work with CalDAV servers and tests various aspects of their protocol handling as well as performance. This widely used implementation supports all of the requirements described in this document and successfully interoperates with the server implementations described above. This implementation is freely distributable under the terms of the Apache License, Version 2.0 [10].

#### 7.7. 2Do

2Do [11] is a standards-compliant calendar client for iOS which uses the CalDAV standard for communication. This implementation supports all of the requirements described in this document and successfully interoperates with the Calendar and Contacts Server implementation described above. This implementation is proprietary and available for purchase from the vendor.

### 8. Security Considerations

The security considerations in [RFC4791] and [RFC4918] apply to this extension. Additionally, servers need to be aware that a client could attack underlying storage by POSTing extremely large

attachments and could attack processing time by uploading a recurring event with a large number of overrides and then repeatedly adding, updating, and deleting attachments.

Malicious content could be introduced into the calendar server by way of a managed attachment, and propagated to many end users via scheduling. Servers SHOULD check managed attachments for malicious or inappropriate content. Upon detecting of such content, servers SHOULD remove the attachment, following the rules described in Section 3.12.5.

## 9. IANA Considerations

### 9.1. Parameter Registrations

This specification defines the following new iCalendar property parameters to be added to the registry defined in Section 8.2.3 of [RFC5545]:

Property Parameter	Status	Reference
SIZE	Current	RFCXXXX, Section 4.1
FILENAME	Current	RFCXXXX, Section 4.2
MANAGED-ID	Current	RFCXXXX, Section 4.3

### 9.2. Message Header Field Registrations

The message header fields below should be added to the Permanent Message Header Field Registry (see [RFC3864]).

#### 9.2.1. Cal-Managed-ID

Header field name: Cal-Managed-ID

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification (Section 5.1)

Related information: none

## 10. Acknowledgments

This specification came about via discussions at the Calendaring and Scheduling Consortium. Thanks in particular to Mike Douglass and Eric York.

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2518] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and D. Jensen, "HTTP Extensions for Distributed Authoring -- WEBDAV", RFC 2518, DOI 10.17487/RFC2518, February 1999, <<https://www.rfc-editor.org/info/rfc2518>>.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <<https://www.rfc-editor.org/info/rfc3864>>.
- [RFC4331] Korver, B. and L. Dusseault, "Quota and Size Properties for Distributed Authoring and Versioning (DAV) Collections", RFC 4331, DOI 10.17487/RFC4331, February 2006, <<https://www.rfc-editor.org/info/rfc4331>>.
- [RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <<https://www.rfc-editor.org/info/rfc4791>>.
- [RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <<https://www.rfc-editor.org/info/rfc4918>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.

- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC6266] Reschke, J., "Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)", RFC 6266, DOI 10.17487/RFC6266, June 2011, <<https://www.rfc-editor.org/info/rfc6266>>.
- [RFC6638] Daboo, C. and B. Desruisseaux, "Scheduling Extensions to CalDAV", RFC 6638, DOI 10.17487/RFC6638, June 2012, <<https://www.rfc-editor.org/info/rfc6638>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7240] Snell, J., "Prefer Header for HTTP", RFC 7240, DOI 10.17487/RFC7240, June 2014, <<https://www.rfc-editor.org/info/rfc7240>>.
- [RFC7538] Reschke, J., "The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect)", RFC 7538, DOI 10.17487/RFC7538, April 2015, <<https://www.rfc-editor.org/info/rfc7538>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 11.2. Informative References

- [RFC5023] Gregorio, J., Ed. and B. de hOra, Ed., "The Atom Publishing Protocol", RFC 5023, DOI 10.17487/RFC5023, October 2007, <<https://www.rfc-editor.org/info/rfc5023>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.



- [RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <<https://www.rfc-editor.org/info/rfc7320>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8144] Murchison, K., "Use of the Prefer Header Field in Web Distributed Authoring and Versioning (WebDAV)", RFC 8144, DOI 10.17487/RFC8144, April 2017, <<https://www.rfc-editor.org/info/rfc8144>>.

### 11.3. URIs

- [1] <https://tools.ietf.org/html/bcp14>
- [2] <http://calendarserver.org/>
- [3] <http://www.apache.org/licenses/LICENSE-2.0.html>
- [4] <http://www.cyrusimap.org/>
- [5] <http://www.cmu.edu/computing/>
- [6] <http://www.cyrusimap.org/>
- [7] <http://www.apple.com/macos/>
- [8] <http://www.busymac.com/busycal/>
- [9] <http://calendarserver.org/wiki/CalDAVTester>
- [10] <http://www.apache.org/licenses/LICENSE-2.0.html>
- [11] <http://www.2doapp.com/>

### Appendix A. Change History (To be removed by RFC Editor before publication)

#### Changes in calext-04:

1. Added text explaining why this document is being published as Informational.
2. Further specified Cal-Managed-ID per Section 8.3.1 of RFC 7231.

3. Specified that the MANAGED-ID parameter value is unique within the scope of the server.
4. Added more text regarding preconditions.
5. Added text about specific DoS attack vectors.
6. Editorial changes from Gren Elliot and Phillip Kewisch.
7. Editorial changes from Adam Roach.
8. Editorial changes from Alexey Melnikov.
9. Added reference to RFC4918.
10. Minor editorial changes.

Changes in calext-03:

1. Changed to Informational based on feedback regarding non-standard method of updating an existing resource.
2. Added references to sub-sections in Overview.
3. Made support for Prefer header field a SHOULD for servers.
4. Expanded recurring event examples to use conditional requests and to include the Expect header field.
5. Minor editorial changes.

Changes in calext-02:

1. Moved "Error Handling" into its own sub-section.
2. Split "Other Client Considerations" into "Processing Time" and "Migrating Calendar Data".

Changes in calext-01:

1. Changed all instances of "header" to "header field".
2. Reworked wording of Prefer header field handling.
3. Switched to recommending 102 (Processing) interim response to keep the client connection alive.

4. Fixed description of Cal-Managed-ID response header field to state that it is also required in responses to successful attachment-update.
5. Minor editorial changes.

Changes in calext-00:

1. Added Murchison as editor.
2. Updated HTTP references to RFC7230 and RFC7231.
3. Updated Prefer header field references to RFC7240.
4. Added Implementation Status section.
5. Minor editorial changes.

Changes in daboo-03:

1. Fixed some examples.
2. Fixed return-representation -> return=representation.
3. Added statement that servers must not allow clients to DELETE attachments directly.
4. Added new preconditions for valid managed-id values.
5. Filled out Access Control section.
6. Allow servers to not support per-instance attachments and advertise that fact to clients.

Changes in daboo-02:

1. MANAGED-ID changes on PUT.
2. MTAG has been removed.
3. Error pre-conditions added.
4. Interaction with WebDAV QUOTA discussed.
5. max-attachment-\* limits added.
6. Updated references.

7. Removed MUST for specific 2xx codes in favor of generic success code.

Changes in daboo-01:

1. Tweaked OPTIONS capability wording.
2. Added section on clients expecting 100-Continue for delayed response.
3. Added text for clean-up and use of HTTP 410 on orphans.
4. Added text on removing "MANAGED-ID" when exporting/importing calendar data.
5. Added protocol examples.
6. Added MTAG property parameter on ATTACH property
7. Added FILENAME property parameter on ATTACH property
8. "id" query parameter is now "managed-id".
9. Use of Cal-Managed-ID header instead of Location header in responses.
10. rid query param MUST contain RECURRENCE-ID without any conversion to UTC (case of floating events).
11. Introduced CALDAV:managed-attachments-server-URL property
12. Made support for Prefer header a MUST for servers.

#### Appendix B. Example Involving Recurring Events

In the following example, the organizer of a recurring meeting makes an unsuccessful attempt to add an agenda (HTML attachment) to the corresponding calendar resource with a conditional request. Note that the client includes both the Expect and Prefer header fields in the request, thereby preventing itself from needlessly sending the attachment data, and requesting that the current resource be returned in the failure response (see Section 3.2 of [RFC8144]).

>> Request <<

```
POST /events/65.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment;filename=agenda.html
Content-Length: xxxx
If-Match: "abcdefg-000"
Expect: 100-continue
Prefer: return=representation
```

>> Final Response <<

HTTP/1.1 412 Precondition Failed  
Content-Type: text/calendar; charset="utf-8"  
Content-Length: yyyy  
Content-Location: https://cal.example.com/events/65.ics  
ETag: "123456789-000-000"

BEGIN:VCALENDAR  
VERSION:2.0  
PRODID:-//Example Corp.//CalDAV Server//EN  
BEGIN:VTIMEZONE  
LAST-MODIFIED:20040110T032845Z  
TZID:America/Montreal  
BEGIN:DAYLIGHT  
DTSTART:20000404T020000  
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4  
TZNAME:EDT  
TZOFFSETFROM:-0500  
TZOFFSETTO:-0400  
END:DAYLIGHT  
BEGIN:STANDARD  
DTSTART:20001026T020000  
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10  
TZNAME:EST  
TZOFFSETFROM:-0400  
TZOFFSETTO:-0500  
END:STANDARD  
END:VTIMEZONE  
BEGIN:VEVENT  
UID:20010712T182145Z-123401@example.com  
DTSTAMP:20120201T203412Z  
DTSTART;TZID=America/Montreal:20120206T100000  
DURATION:PT1H  
RRULE:FREQ=WEEKLY  
SUMMARY:Planning Meeting  
ORGANIZER:mailto:cyrus@example.com  
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@example.com  
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@example.com  
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@example.com  
END:VEVENT  
END:VCALENDAR

The organizer of a recurring meeting successfully adds an agenda (HTML attachment) to the corresponding calendar resource. Attendees

of the meeting are granted read access to the newly created attachment resource. Their own copy of the meeting is updated to include the new ATTACH property pointing to the attachment resource and they are notified of the change via their scheduling inbox.

>> Request <<

```
POST /events/65.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment;filename=agenda.html
Content-Length: xxxx
If-Match: "123456789-000-000"
Expect: 100-continue
Prefer: return=representation
```

>> Interim Response <<

```
HTTP/1.1 100 Continue
```

>> Request Body <<

```
<html>
  <body>
    <h1>Agenda</h1>
    <p>As usual</p>
  </body>
</html>
```

>> Final Response <<

HTTP/1.1 201 Created  
Content-Type: text/calendar; charset="utf-8"  
Content-Length: yyyy  
Content-Location: https://cal.example.com/events/65.ics  
ETag: "123456789-000-111"  
Cal-Managed-ID: 97S

BEGIN:VCALENDAR  
VERSION:2.0  
PRODID:-//Example Corp.//CalDAV Server//EN  
BEGIN:VTIMEZONE  
LAST-MODIFIED:20040110T032845Z  
TZID:America/Montreal  
BEGIN:DAYLIGHT  
DTSTART:20000404T020000  
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4  
TZNAME:EDT  
TZOFFSETFROM:-0500  
TZOFFSETTO:-0400  
END:DAYLIGHT  
BEGIN:STANDARD  
DTSTART:20001026T020000  
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10  
TZNAME:EST  
TZOFFSETFROM:-0400  
TZOFFSETTO:-0500  
END:STANDARD  
END:VTIMEZONE  
BEGIN:VEVENT  
UID:20010712T182145Z-123401@example.com  
DTSTAMP:20120201T203412Z  
DTSTART;TZID=America/Montreal:20120206T100000  
DURATION:PT1H  
RRULE:FREQ=WEEKLY  
SUMMARY:Planning Meeting  
ORGANIZER:mailto:cyrus@example.com  
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@example.com  
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@example.com  
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@example.com  
ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;  
FILENAME=agenda.html:https://cal.example.com/attach/65/34X22R  
END:VEVENT  
END:VCALENDAR



The organizer has a more specific agenda for the 20th of February meeting. It is added to that particular instance of the meeting by specifying the rid parameter. Note that an overridden instance is created with the RECURRENCE-ID property value matching the value of the "rid" query parameter in the request. Also note that the server takes significant time to complete the request and notifies the client accordingly.

>> Request <<

```
POST /events/65.ics?action=attachment-add&rid=20120220T100000 HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition: attachment;filename=agenda0220.html
Content-Length: xxxx
If-Match: "123456789-000-111"
Expect: 100-continue
Prefer: return=representation
```

>> Interim Response <<

```
HTTP/1.1 100 Continue
```

>> Request Body <<

```
<html>
  <body>
    <h1>Agenda</h1>
    <p>Something different, for a change</p>
  </body>
</html>
```

>> Interim Response <<

```
HTTP/1.1 102 Processing
```

>> Final Response <<

```
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://cal.example.com/events/65.ics
ETag: "123456789-000-222"
Cal-Managed-ID: 33225
```

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VTIMEZONE
LAST-MODIFIED:20040110T032845Z
TZID:America/Montreal
BEGIN:DAYLIGHT
DTSTART:20000404T020000
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
TZNAME:EDT
TZOFFSETFROM:-0500
TZOFFSETTO:-0400
END:DAYLIGHT
BEGIN:STANDARD
DTSTART:20001026T020000
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
TZNAME:EST
TZOFFSETFROM:-0400
TZOFFSETTO:-0500
END:STANDARD
END:VTIMEZONE
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
DTSTAMP:20120201T203412Z
DTSTART;TZID=America/Montreal:20120206T100000
DURATION:PT1H
RRULE:FREQ=WEEKLY
SUMMARY:Planning Meeting
ORGANIZER:mailto:cyrus@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@exampl
e.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exam
ple.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@exa
mple.com
ATTACH;MANAGED-ID=97S;FMPTYPE=text/html;SIZE=xxxx;
FILENAME=agenda.html:https://cal.example.com/attach/65/34X22R
END:VEVENT
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
RECURRENCE-ID;TZID=America/Montreal:20120220T100000
DTSTAMP:20120201T203412Z
DTSTART;TZID=America/Montreal:20120220T100000
DURATION:PT1H
SUMMARY:Planning Meeting
ORGANIZER:mailto:cyrus@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@example.
com
```

```
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exampl
e.com
ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@examp
le.com
ATTACH;MANAGED-ID=33225;FMPTYPE=text/html;SIZE=xxxx;
  FILENAME=agenda0220.html:https://cal.example.com/attach/65/FGZ225
END:VEVENT
END:VCALENDAR
```

## Authors' Addresses

Cyrus Daboo  
Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
USA

Email: [cyrus@daboo.name](mailto:cyrus@daboo.name)  
URI: <http://www.apple.com/>

Arnaud Quillaud  
Oracle Corporation  
180, Avenue de l'Europe  
Saint Ismier cedex 38334  
France

Email: [arnaud.quillaud@oracle.com](mailto:arnaud.quillaud@oracle.com)  
URI: <http://www.oracle.com/>

Kenneth Murchison (editor)  
FastMail US LLC  
1429 Walnut St, Suite 1201  
Philadelphia, PA 19102  
USA

Email: [murch@fastmailteam.com](mailto:murch@fastmailteam.com)  
URI: <http://www.fastmail.com/>

Network Working Group  
Internet-Draft  
Updates: 5545 (if approved)  
Intended status: Standards Track  
Expires: September 27, 2021

M. Douglass  
Bedework  
March 26, 2021

Event Publishing Extensions to iCalendar  
draft-ietf-calext-eventpub-extensions-19

Abstract

This specification updates RFC5545 by introducing a number of new iCalendar properties and components which are of particular use for event publishers and in social networking.

This specification also defines a new STRUCTURED-DATA property for iCalendar RFC5545 to allow for data that is directly pertinent to an event or task to be included with the calendar 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 September 27, 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. Conventions Used in This Document . . . . .	3
1.2. Terms Used in This Document . . . . .	3
2. Components and properties . . . . .	4
3. Typed References . . . . .	4
3.1. Use Cases . . . . .	5
3.1.1. Piano Concert Performance . . . . .	5
3.1.2. Itineraries . . . . .	5
3.1.2.1. Reserving facilities . . . . .	6
4. Modifications to Calendar Components . . . . .	6
5. New Property Parameters . . . . .	7
5.1. Order . . . . .	7
5.2. Schema . . . . .	8
5.3. Derived . . . . .	9
6. New Properties . . . . .	10
6.1. Location Type . . . . .	10
6.2. Participant Type . . . . .	11
6.3. Resource Type . . . . .	13
6.4. Calendar Address . . . . .	14
6.5. Styled-Description . . . . .	14
6.6. Structured-Data . . . . .	16
7. New Components . . . . .	19
7.1. Participant . . . . .	19
7.1.1. Schedulable Participant . . . . .	21
7.2. Location . . . . .	22
7.3. Resource . . . . .	23
8. Extended examples . . . . .	25
8.1. Example 1 . . . . .	25
8.2. Example 2 . . . . .	26
9. Security Considerations . . . . .	26
9.1. URIs . . . . .	26
9.2. Malicious Content . . . . .	27
9.3. HTML Content . . . . .	27
10. Privacy Considerations . . . . .	27
10.1. Tracking . . . . .	27
10.2. Revealing Locations . . . . .	27
11. IANA Considerations . . . . .	28
11.1. Additional iCalendar Registrations . . . . .	28
11.1.1. Properties . . . . .	28
11.1.2. Parameters . . . . .	28
11.1.3. Components . . . . .	29
11.2. New Registration Tables . . . . .	29

11.2.1. Participant Types . . . . .	29
11.2.2. Resource Types . . . . .	30
12. Acknowledgements . . . . .	30
13. Normative References . . . . .	30
Appendix A. Open issues . . . . .	32
Appendix B. Change log . . . . .	32
Author's Address . . . . .	35

## 1. Introduction

The currently existing iCalendar standard [RFC5545] lacks useful methods for referencing additional, external information relating to calendar components. Additionally there is no standard way to provide rich text descriptions or meta-data associated with the event.

Current practice is to embed this information as links in the description or to add non-standard properties as defined in [RFC5545] section 3.8.8.2.

This document updates [RFC5545] to define a number of properties and components referencing such external information that can provide additional information about an iCalendar component. The intent is to allow interchange of such information between applications or systems (e.g., between clients, between client and server, and between servers). Formats such as vCard [RFC2426] are likely to be most useful to the receivers of such events as they may be used in other applications - such as address books.

### 1.1. 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.

### 1.2. Terms Used in This Document

**Event:** When the (perhaps with a capitalised 'E') word 'event' is used we are referring to gatherings, formal or informal. For example a sports event, a party or a concert.

**Social Calendaring:** Historically, calendar data and scheduling has been heavily biased towards meetings in a corporate environment. Some of the features defined in this document are to support a more informal, i.e. social, model. For example, we may want to record who is participating in a public event.

## 2. Components and properties

Previous extensions to the calendaring standards have been largely restricted to the addition of properties or parameters. This is partly because iCalendar libraries had trouble handling components nested deeper than those defined in [RFC5545].

In a break with this 'tradition' this specification defines a number of components rather than properties. This is a better match for the way [W3C.REC-xml-20081126] and JSON [RFC8259] handle such structures and allows richer definitions.

It also allows for the addition of extra properties inside the components and resolves some of the problems of trying to add detailed information as a parameter.

## 3. Typed References

The properties and components defined here can all reference external meta-data which may be used by applications to provide further information to users. By providing type information, clients and servers are able to discover interesting references and make use of them, perhaps for indexing or the presenting of additional related information for the user.

As always, clients should exercise caution in following references to external data.

The [RFC5545] LOCATION property provides only an unstructured single text value for specifying the location where an event (or task) will occur. This is inadequate for use cases where structured location information (e.g. address, region, country, postal code) is required or preferred, and limits widespread adoption of iCalendar in those settings.

Using the VLOCATION component, rich information about multiple locations can be communicated in a STRUCTURED-DATA property, for example, address, region, country, postal code as well as other information such as parking availability, nearby restaurants and the venue. Servers and clients can retrieve the objects when storing the event and use them to index by geographic location.

When a calendar client receives a calendar component it can search the set of locations looking for those of particular interest. The LOCATION-TYPE property and STRUCTURED-DATA FMTTYPE parameter, if supplied, can be used to help the selection.

The PARTICIPANT component is designed to handle common use cases in event publication. It is generally important to provide information about the organizers of such events. Sponsors wish to be referenced in a prominent manner. In social calendaring it is often important to identify the active participants in the event, for example a school sports team, and the inactive participants, for example the parents.

The PARTICIPANT component can be used to provide useful extra data about an attendee. For example a location inside the PARTICIPANT gives the actual location of a remote attendee. (But see the note about privacy.)

Alternatively the PARTICIPANT component can be used to provide a reference - perhaps the address for mailing lists.

### 3.1. Use Cases

The main motivation for these changes has been event publication but there are opportunities for use elsewhere. The following use cases will describe some possible scenarios.

#### 3.1.1. Piano Concert Performance

In putting together a concert there are many participants: piano tuner, performer, stage hands etc. In addition there are sponsors and various contacts to be provided. There will also be a number of related locations. A number of events can be created, all of which relate to the performance in different ways.

There may be an iTIP [RFC5546] meeting request for the piano tuner who will arrive before the performance. Other members of staff may also receive meeting requests.

An event can also be created for publication which will have a PARTICIPANT component for the pianist providing a reference to vCard [RFC2426] information about the performer. This event would also hold information about parking, local subway stations and the venue itself. In addition, there may be sponsorship information for sponsors of the event and perhaps paid sponsorship properties essentially advertising local establishments.

#### 3.1.2. Itineraries

These additions also provide opportunities for the travel industry. When booking a flight the PARTICIPANT component can be used to provide references to businesses at the airports and to car hire businesses at the destination.



The embedded location information can guide the traveler at the airport or to their final destination. The contact information can provide detailed information about the booking agent, the airlines, car hire companies and the hotel.

#### 3.1.2.1. Reserving facilities

For a meeting, the size of a room and the equipment needed depends to some extent on the number of attendees actually in the room.

A meeting may have many attendees none of which are co-located. The current ATTENDEE property does not allow for the addition of such meta-data. The PARTICIPANT component allows attendees to specify their location.

### 4. Modifications to Calendar Components

The following changes to the syntax defined in iCalendar [RFC5545] are made here. New elements are defined in subsequent sections.

```
; Addition of PARTICIPANT, VLOCATION and VRESOURCE
; as valid components
eventc      = "BEGIN" ":" "VEVENT" CRLF
              eventprop *alarmc *participantc *locationc *resourcec
              "END" ":" "VEVENT" CRLF

; Addition of properties STYLED-DESCRIPTION and STRUCTURED-DATA
eventprop   =/ *styleddescription
              *sdataprop

; Addition of PARTICIPANT, VLOCATION and VRESOURCE
; as valid components
todoc       = "BEGIN" ":" "VTODO" CRLF
              todoprop *alarmc *participantc *locationc *resourcec
              "END" ":" "VTODO" CRLF

; Addition of properties STYLED-DESCRIPTION, STRUCTURED-DATA
todoprop    =/ *styleddescription
              *sdataprop

; Addition of PARTICIPANT, VLOCATION and VRESOURCE
; as valid components
journalc    = "BEGIN" ":" "VJOURNAL" CRLF
              jourprop *participantc *locationc *resourcec
              "END" ":" "VJOURNAL" CRLF

; Addition of properties STYLED-DESCRIPTION, STRUCTURED-DATA
jourprop    =/ *styleddescription
              *sdataprop

; Addition of PARTICIPANT, VLOCATION and VRESOURCE
; as valid components
freebusyc   = "BEGIN" ":" "VFREEBUSY" CRLF
              fbprop *participantc *locationc *resourcec
              "END" ":" "VFREEBUSY" CRLF

; Addition of property STYLED-DESCRIPTION
fbprop      =/ *styleddescription
```

## 5. New Property Parameters

### 5.1. Order

Parameter name: ORDER

Purpose: To define ordering for the associated property.

**Format Definition:**

This parameter is defined by the following notation:

```
orderparam    = "ORDER" "=" integer  
                ;           Must be greater than or equal to 1
```

**Description:** The ORDER parameter is OPTIONAL and is used to indicate the relative ordering of the corresponding instance of a property. Its value MUST be an integer greater than or equal to 1 that specifies the order with 1 being the first in the ordering.

When the parameter is absent, the default MUST be to interpret the property instance as being ordered last, that is, the property will appear after any other instances of the same property with any value of ORDER.

When any ORDER parameters have the same value all the associated properties appear as a group within which there is no defined order.

Note that the value of this parameter is to be interpreted only in relation to values assigned to other corresponding instances of the same property in the same entity.

This parameter MUST NOT be applied to a property that does not allow multiple instances.

**Example uses:** The ORDER may be applied to the PARTICIPANT-TYPE property to indicate the relative importance of the participant, for example as a sponsor or a performer. For example, ORDER=1 could define the principal performer or soloist.

## 5.2. Schema

**Parameter Name:** SCHEMA

**Purpose:** To specify the schema used for the content of a "STRUCTURED-DATA" property value.

**Format Definition:**

This parameter is defined by the following notation:

```
schemaparam    = "SCHEMA" "=" DQUOTE uri DQUOTE
```

**Description:** This property parameter SHOULD be specified on "STRUCTURED-DATA" properties. When present it provides

identifying information about the nature of the content of the corresponding "STRUCTURED-DATA" property value. This can be used to supplement the media type information provided by the "FMCTYPE" parameter on the corresponding property.

Example:

STRUCTURED-DATA;FMTTYPE=application/json;  
SCHEMA="https://schema.org/FlightReservation";  
ENCODING=BASE64;VALUE=BINARY:ICAgIDxzyJ3pcHQgdHlwZT0iYXBwbGljYXRpb24vbGQranNvbiI+CiAgICB7CiAgICAgICJAy29udGV4dCI6CjodHRwOi8vc2NoZWlhLm9yZyIsCiAgICAgICJAdHlwZSI6ICJBGbGlnaHRSZXNlcnZhdGlvbkiIsCiAgICAgICJyZXNlcnZhdGlvbklkIjoGIllJSVJM0UCisCiAgICAgICJCyzXNlcnZhdGlbn0YXRlcyl6ICJoHRwOiwvc2NoZWlhLm9yZSZNlcnZhdGlbnkbWbmZpcmmlzCisiAgiAgICgicWYNXnzW5nZXJQcmlycmcl0eVN0YXRlcyl6ICJGYXN0IFRyYWNIiwKICAgICAgInBhc3NbmdldClnlcXVlbmNiLnVtVmVyIjoGIkkFCQZeMyIsCiAgICAgICJzZWNleml0eVNjcmbVblmluZyI6ICJUUEUegUHJlQ2hlY2siLAogICAgaCAidW5kZXJOYWllIjoGewogICAgICAgICJAdHlwZSI6ICJQZXJzb24iLAogICAgaCAgICJuYWllIjoGIkvZYSBHcmVlbikIKICAgICAgfSwKICAgICAgInJlc2Vydmc0aW9uUm9yIjoGewogICAgICAgICJAdHlwZSI6ICJBGbGlnaHQiLAogICAgaCAgICAJmbGlnaHRoDWlIZXIiOiAiivUEExMTAiLAogICAgaCAgICAJwcml0eWRlcii6IHsKICAgICAgICAgICJAdHlwZSI6ICJBaxJsaw5lliwwKICAgICAgICAgICJuYWllIjoGIknVnbNRpbmVudGFsIiwKICAgICAgICAgICJPYXRhQ29kZSI6ICJDdyIsCiAgICAgaCAgICAiaUYyNm9hcmRpbdQbzxpY3kiOiAiAhRODOL3njaGVtYS5vcmeWm9uZUUyYXJkaW5nUG9saWN5IgogICAgICAgICAIH0scIAgaCAgICAnInlbGxlciI6IHsKICAgICAgICAgICJAdHlwZSI6ICJBaxJsaw5lliwwKICAgICAgICAgICJuYWllIjoGIllVuaXRlcziScIAgaCAgICAgICAiaWF0YUNvZGUioiAiivUEEiCiAgICAgaCAgfSwKICAgICAgICAiazGVwyXJ0dXJlQWlycG9ydCI6IHsKICAgICAgICAgICJAdHlwZSI6ICJBaxJwb3J0IiwKICAgICAgICAgICJuYWllIjoGIllNhbiBGcmFuY2lyZ28qWilycG9ydCIscIAgaCAgICAgICAiaWF0YUNvZGUioiAiIU0ZIgoGAgaCAgICAH0scIAgaCAgaCAgImRlcGFydHVyZVRpbWUiOiAiAmJAxNy0wMy0wNFQyMDoxNTowMCM0wODowMCIsCiAgICAgICAgICImFYcmllYWxBaxJwb3J0IjoGewogICAgICAgaCAgIkBOEXBlIjoGIkFpcnBvenQiLAogICAgaCAgICAgICIm5hbWUiOiAiasm9obiBgLiBLZW5fuZRWSIEluDGvybmFOAw9uYUwgQWlycG9ydCIscIAgaCAgaCAgICAiaWF0YUNvZGUioiAiskZLItoGaCAgICAGIHOscIAgaCAgaCAgICImFYcmllYWBaxNWllIjoGIjjIMTctMDMtMDVMUDY6MZAMDATMDUMDAiCiAgICAgaCAgIHOskKAgaCAgIDwvc2NyaXBOPg==

### 5.3. Derived

Parameter Name: DERIVED

Purpose: To specify that the value of the associated property is derived from some other property value or values.

**Format Definition:**

This parameter is defined by the following notation:

```
derivedparam    = "DERIVED" "=" ("TRUE" / "FALSE")  
; Default is FALSE
```

**Description:** This property parameter MAY be specified on any property when the value is derived from some other property or properties. When present with a value of TRUE clients MUST NOT update the property.

As an example, if a STYLED-DESCRIPTION property is present with FMTTYPE="application/rtf" then there may be an additional STYLED-DESCRIPTION property with FMTTYPE="text/html" and DERIVED=TRUE and a value created from the rtf value.

**Example:**

```
STYLED-DESCRIPTION;FMTTYPE=text/html;  
DERIVED=TRUE:<html>...</html>
```

## 6. New Properties

This specification makes use of the NAME property which is defined in [RFC7986]

### 6.1. Location Type

**Property name:** LOCATION-TYPE

**Purpose:** To specify the type(s) of a location.

**Value type:** The value type for this property is TEXT. The allowable values are defined below.

**Description:** This property MAY be specified in VLOCATION components and provides a way to differentiate multiple locations. For example, it allows event producers to provide location information for the venue and the parking.

**Format Definition:**

This property is defined by the following notation:

```
loctype          = "LOCATION-TYPE" loctypeparam ":"
                  text *("," text)
                  CRLF
```

```
loctypeparam     = *(";" other-param)
```

Multiple values may be used if the location has multiple purposes, for example a hotel and a restaurant.

Values for this parameter are taken from the values defined in [RFC4589] section 3. New location types SHOULD be registered in the manner laid down in section 5 of that specification.

## 6.2. Participant Type

Property name: PARTICIPANT-TYPE

Purpose: To specify the type of participant.

Value type: The value type for this property is TEXT. The allowable values are defined below.

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

Conformance: This property MUST be specified once within a PARTICIPANT component.

Description: This property defines the type of participation in events or tasks. Participants can be individuals or organizations, for example a soccer team, the spectators, or the musicians.

Format Definition:

This property is defined by the following notation:

```
participanttype = "PARTICIPANT-TYPE" partvalueparam ":"
                  partvalue CRLF

partvalue       = ("ACTIVE"
                   / "INACTIVE"
                   / "SPONSOR"
                   / "CONTACT"
                   / "BOOKING-CONTACT"
                   / "EMERGENCY-CONTACT"
                   / "PUBLICITY-CONTACT"
                   / "PLANNER-CONTACT"
                   / "PERFORMER"
                   / "SPEAKER"
                   / iana-token) ; Other IANA-registered
                                ; values

partvalueparam  = *(";" other-param)
```

Example:

The following is an example of this property:

```
PARTICIPANT-TYPE:SPEAKER
```

The registered values for the PARTICIPANT-TYPE property have the meanings described here:

ACTIVE: A participant taking an active role - for example a team member.

INACTIVE: A participant taking an inactive role - for example an audience member.

SPONSOR: A sponsor of the event. The ORDER parameter may be used with this participant type to define the relative order of multiple sponsors.

CONTACT: Contact information for the event. The ORDER parameter may be used with this participant type to define the relative order of multiple contacts.

BOOKING-CONTACT: Contact information for reservations or payment

EMERGENCY-CONTACT: Contact in case of emergency

PUBLICITY-CONTACT: Contact for publicity

PLANNER-CONTACT: Contact for the event planner or organizer

PERFORMER: A performer - for example the soloist or the accompanist.  
The ORDER parameter may be used with this participant type to define the relative order of multiple performers. For example, ORDER=1 could define the principal performer or soloist.

SPEAKER: Speaker at an event

### 6.3. Resource Type

Property name: RESOURCE-TYPE

Purpose: To specify the type of resource.

Value type: The value type for this property is TEXT. The allowable values are defined below.

Format Definition:

This property is defined by the following notation:

```
restypeprop  = "RESOURCE-TYPE" restypeparam ":"  
               restypevalue CRLF  
  
restypevalue = ("ROOM"  
               / "PROJECTOR"  
               / "REMOTE-CONFERENCE-AUDIO"  
               / "REMOTE-CONFERENCE-VIDEO"  
               / iana-token) ; Other IANA-registered  
               ; values  
  
restypeparam = *(";" other-param)
```

Description: This property MAY be specified in VRESOURCE components and provides a way to differentiate multiple resources.

The registered values are described below. New resource types SHOULD be registered in the manner laid down in this specification.

ROOM: A room for the event/meeting.

PROJECTOR: Projection equipment.

REMOTE-CONFERENCE-AUDIO: Audio remote conferencing facilities.

REMOTE-CONFERENCE-VIDEO: Video remote conferencing facilities.



#### 6.4. Calendar Address

Property name: CALENDAR-ADDRESS

Purpose: To specify the calendar address for a participant.

Value type: CAL-ADDRESS

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

Conformance: This property MAY be specified once within a PARTICIPANT component.

Description: This property provides a calendar user address for the participant. If there is an ATTENDEE property with the same value then the participant is schedulable.

Format Definition:

This property is defined by the following notation:

```
calendaraddress = "CALENDAR-ADDRESS" caladdressparam ":"  
                cal-address CRLF  
  
caladdressparam = *(";" other-param)
```

#### 6.5. Styled-Description

Property name: STYLED-DESCRIPTION

Purpose: This property provides for one or more rich-text descriptions to replace that provided by the DESCRIPTION property.

Value type: There is no default value type for this property. The value type can be set to URI or TEXT. Other text-based value types can be used when defined in the future. Clients MUST ignore any properties with value types they do not understand.

Property Parameters: IANA-registered, non-standard, id, alternate text representation, format type, derived and language property parameters can be specified on this property.

Conformance: The property can be specified multiple times in the "VEVENT", "VTODO", "VJOURNAL", "VFREEBUSY", "PARTICIPANT", or "VALARM" calendar components.

If it does appear more than once there MUST be exactly one instance of the property with no DERIVED parameter or DERIVED=FALSE. All others MUST have DERIVED=TRUE.

Additionally, if there is one or more STYLED-DESCRIPTION property then the DESCRIPTION property should be either absent or have the parameter DERIVED=TRUE.

**Description:** This property supports rich-text descriptions, for example HTML. Event publishers typically wish to provide more and better formatted information about the event.

This property is used in the "VEVENT" and "VTODO" to capture lengthy textual descriptions associated with the activity. This property is used in the "VJOURNAL" calendar component to capture one or more textual journal entries. This property is used in the "VALARM" calendar component to capture the display text for a DISPLAY category of alarm, and to capture the body text for an EMAIL category of alarm. In the PARTICIPANT component it provides a detailed description of the participant.

VALUE=TEXT is used to provide rich-text inline as the property value.

VALUE=URI is used to provide a link to rich-text content which is expected to be displayed inline as part of the event.

In either case the DESCRIPTION property should be absent or contain a plain text rendering of the styled text.

Applications MAY attempt to guess the media type of the resource via inspection of its content if and only if the media type of the resource is not given by the "FMTTYPE" parameter. If the media type remains unknown, calendar applications SHOULD treat it as type "text/html" and process the content as defined in [W3C.REC-html51-20171003]

Multiple STYLED-DESCRIPTION properties may be used to provide different formats or different language variants. However all but one MUST have DERIVED=TRUE.

**Format Definition:**

This property is defined by the following notation:

```

styleddescription = "STYLED-DESCRIPTION" styleddescparam ":"
                    styleddescval CRLF

styleddescparam   = *(
                    ; The following is REQUIRED,
                    ; but MUST NOT occur more than once.
                    ;
                    (";" "VALUE" "=" ("URI" / "TEXT")) /
                    ;
                    ; The following are OPTIONAL,
                    ; but MUST NOT occur more than once.
                    ;
                    (";" altrepparam) / (";" languageparam) /
                    (";" fmttypeparam) / (";" derivedparam) /
                    ;
                    ; the following is OPTIONAL
                    ; and MAY occur more than once
                    ;
                    (";" other-param)
                    )

styleddescval     = ( uri / text )
;Value MUST match value type

```

Example:

The following is an example of this property. It points to an html description.

```
STYLED-DESCRIPTION;VALUE=URI:http://example.org/desc001.html
```

## 6.6. Structured-Data

Property Name: STRUCTURED-DATA

Purpose: This property specifies ancillary data associated with the calendar component.

Value Type: There is no default value type for this property. The value type can be set to TEXT, BINARY or URI

Property Parameters: IANA-registered, non-standard, inline encoding and value data type property parameters can be specified on this property. The format type and schema parameters can be specified on this property and MUST be present for text or inline binary encoded content information.

Conformance: This property can be specified multiple times in an iCalendar object. Typically it would be used in "VEVENT", "VTODO" or "VJOURNAL" calendar components.

Description: The existing properties in iCalendar cover key elements of events and tasks such as start time, end time, location, summary, etc. However, different types of events often have other specific "fields" that it is useful to include in the calendar data. For example, an event representing an airline flight could include the airline, flight number, departure and arrival airport codes, check-in and gate-closing times etc. As another example, a sporting event might contain information about the type of sport, the home and away teams, the league the teams are in, information about nearby parking, etc.

This property is used to specify ancillary data in some structured format either directly (inline) as a "TEXT" or "BINARY" value or as a link via a "URI" value.

Rather than define new iCalendar properties for the variety of event types that might occur, it would be better to leverage existing schemas for such data. For example, schemas available at <https://schema.org> include different event types. By using standard schemas, interoperability can be improved between calendar clients and non-calendar systems that wish to generate or process the data.

This property allows the direct inclusion of ancillary data whose schema is defined elsewhere. This property also includes parameters to clearly identify the type of the schema being used so that clients can quickly and easily spot what is relevant within the calendar data and present that to users or process it within the calendaring system.

iCalendar does support an "ATTACH" property which can be used to include documents or links to documents within the calendar data. However, that property does not allow data to be included as a "TEXT" value (a feature that "STRUCTURED-DATA" does allow), plus attachments are often treated as "opaque" data to be processed by some other system rather than the calendar client. Thus the existing "ATTACH" property is not sufficient to cover the specific needs of inclusion of schema data. Extending the "ATTACH" property to support a new value type would likely cause interoperability problems. Additionally some implementations manage attachments by stripping them out and replacing with a link to the resource. Thus a new property to support inclusion of schema data is warranted.

## Format Definition:

This property is defined by the following notation:

```
sdataprop      = "STRUCTURED-DATA" sdataparam
                  (
                    ";" "VALUE" "=" "TEXT"
                    ":" text
                  ) /
                  (
                    ";" "ENCODING" "=" "BASE64"
                    ";" "VALUE" "=" "BINARY"
                    ":" binary
                  ) /
                  (
                    ";" "VALUE" "=" "URI"
                    ":" uri
                  )
                  CRLF

sdataparam      = *(
                    ;
                    ; The following is OPTIONAL for a URI value,
                    ; REQUIRED for a TEXT or BINARY value,
                    ; and MUST NOT occur more than once.
                    ;
                    (";" fmttypeparam) /
                    (";" schemaparam) /
                    ;
                    ; The following is OPTIONAL,
                    ; and MAY occur more than once.
                    ;
                    (";" other-param)
                    ;
                  )
```

Example: The following is an example of this property:

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

## 7. New Components

### 7.1. Participant

Component name: PARTICIPANT

Purpose: This component provides information about a participant in an event or task.

Conformance: This component can be specified multiple times in a "VEVENT", "VTODO", "VJOURNAL" or "VFREEBUSY" calendar component.

Description: This component provides information about a participant in a calendar component. A participant may be an attendee in a scheduling sense and the ATTENDEE property may be specified in addition. Participants can be individuals or organizations, for example a soccer team, the spectators or the musicians.

STRUCTURED-DATA properties if present may refer to definitions of the participant - such as a vCard.

The CALENDAR-ADDRESS property if present will provide a cal-address. If an ATTENDEE property has the same value the participant is considered schedulable. The PARTICIPANT component can be used to contain additional meta-data related to the attendee.

Format Definition:

This component is defined by the following notation:

```
participantc = "BEGIN" ":" "PARTICIPANT" CRLF
               partprop *locationc *resourcec
               "END" ":" "PARTICIPANT" CRLF

partprop      = *(
               ;
               ; The following are REQUIRED,
               ; but MUST NOT occur more than once.
               ;
               participanttype / uid /
               ;
               ; The following are OPTIONAL,
               ; but MUST NOT occur more than once.
               ;
               calendaraddress / created / description / dtstamp /
               geo / last-mod / priority / seq /
               status / summary / url /
               ;
               ; The following are OPTIONAL,
               ; and MAY occur more than once.
               ;
               attach / categories / comment /
               contact / location / rstatus / related /
               resources / strucloc / strucres / styleddescription /
               sdataprop / iana-prop
               ;
               )
```

Note: When the PRIORITY is supplied it defines the ordering of PARTICIPANT components with the same value for the PARTICIPANT-TYPE property.

Privacy Issues: When a LOCATION is supplied it provides information about the location of a participant at a given time or times. This may represent an unacceptable privacy risk for some participants. User agents MUST NOT broadcast this information without the express permission of the participants whose location would be exposed. For further comments see Section 10

Example:

The following is an example of this component. It contains a STRUCTURED-DATA property which points to a vCard providing information about the event participant.

```
BEGIN:PARTICIPANT
UID: em9lQGZvb2GFtcGx1LmNvbQ
PARTICIPANT-TYPE:PERFORMER
STRUCTURED-DATA;VALUE=URI:
  http://dir.example.com/vcard/aviolinist.vcf
END:PARTICIPANT
```

Example:

The following is an example for the primary contact.

```
BEGIN:PARTICIPANT
UID: em9lQGZvb2GFtcGx1LmNvbQ
STRUCTURED-DATA;VALUE=URI;
  http://dir.example.com/vcard/contacts/contact1.vcf
PARTICIPANT-TYPE:CONTACT
DESCRIPTION:A contact
END:PARTICIPANT
```

Example:

The following is an example for a participant with contact and location.

```
BEGIN:PARTICIPANT
UID: em9lQGZvb2GFtcGx1LmNdrt
STRUCTURED-DATA;VALUE=URI;
  http://dir.example.com/vcard/contacts/my-card.vcf
PARTICIPANT-TYPE:SPEAKER
DESCRIPTION:A participant
BEGIN:VLOCATION
UID:123456-abcdef-98765432
NAME:My home location
STRUCTURED-DATA;VALUE=URI:
  http://dir.example.com/addresses/my-home.vcf
END:VLOCATION
END:PARTICIPANT
```

#### 7.1.1. Schedulable Participant

A PARTICIPANT component may represent someone or something that needs to be scheduled as defined for ATTENDEE in [RFC5545] and [RFC5546]. The PARTICIPANT component may also represent someone or something that is NOT to receive scheduling messages.



For backwards compatibility with existing clients and servers when used to schedule events and tasks the ATTENDEE property MUST be used to specify the scheduling parameters as defined for that property.

For other, future uses the CALENDAR-ADDRESS property MUST be used to specify those parameters.

A PARTICIPANT component is defined to be schedulable if

- o It contains a CALENDAR-ADDRESS property
- o That property value is the same as the value for an ATTENDEE property.

If both of these conditions apply then the participant defined by the value of the URL property will take part in scheduling operations as defined in [RFC5546].

An appropriate use for the PARTICIPANT component in scheduling would be to store SEQUENCE and DTSTAMP properties associated with replies from each ATTENDEE. A LOCATION property within the PARTICIPANT component might allow better selection of meeting times when participants are in different timezones.

## 7.2. Location

Component name: VLOCATION

Purpose: This component provides rich information about the location of an event using the structured data property or optionally a plain text typed value.

Conformance: This component can be specified multiple times in a "VEVENT", "VTODO", "VJOURNAL", "VFREEBUSY" or "PARTICIPANT" calendar component.

Description: There may be a number of locations associated with an event. This component provides detailed information about a location.

When used in a component the value of this property provides information about the event venue or of related services such as parking, dining, stations etc..

STRUCTURED-DATA properties if present may refer to representations of the location - such as a vCard.

Format Definition:

This component is defined by the following notation:

```
locationc      = "BEGIN" ":" "VLOCATION" CRLF
                  locprop
                  "END" ":" "VLOCATION" CRLF

locprop        = *(
                  ;
                  ; The following are REQUIRED,
                  ; but MUST NOT occur more than once.
                  ;
                  uid /
                  ;
                  ; The following are OPTIONAL,
                  ; but MUST NOT occur more than once.
                  ;
                  description / geo / loctype / name /
                  ;
                  ; The following are OPTIONAL,
                  ; and MAY occur more than once.
                  ;
                  sdataprop / iana-prop
```

The NAME property is defined in [RFC7986]

Example:

The following is an example of this component. It points to a venue.

```
BEGIN:VLOCATION
UID:123456-abcdef-98765432
NAME:The venue
STRUCTURED-DATA;VALUE=URI:
  http://dir.example.com/venues/big-hall.vcf
END:VLOCATION
```

### 7.3. Resource

Component name: VRESOURCE

**Purpose:** This component provides a typed reference to external information about a resource or optionally a plain text typed value. Typically a resource is anything that might be required or used by a calendar entity and possibly has a directory entry.

**Conformance:** This component can be specified multiple times in a "VEVENT", "VTODO", "VJOURNAL", "VFREEBUSY" or "PARTICIPANT" calendar component.

Description: When used in a component this component provides information about resources used for the event such as rooms, projectors, conferencing capabilities.

The RESOURCE-TYPE value registry provides a place in which resource types may be registered.

STRUCTURED-DATA properties if present may refer to representations of the resource - such as a vCard.

#### Format Definition:

This component is defined by the following notation:

```
resourcec      = "BEGIN" ":" "VRESOURCE" CRLF
                  resprop
                  "END" ":" "VRESOURCE" CRLF

resprop        = *(
                  ;
                  ; The following are REQUIRED,
                  ; but MUST NOT occur more than once.
                  ;
                  uid /
                  ;
                  ; The following are OPTIONAL,
                  ; but MUST NOT occur more than once.
                  ;
                  description / geo / name / restype /
                  ;
                  ; The following are OPTIONAL,
                  ; and MAY occur more than once.
                  ;
                  sdataprop / iana-prop
```

The NAME property is defined in [RFC7986]

#### Example:

The following is an example of this component. It refers to a projector.

```
BEGIN:VRESOURCE
UID:456789-abcdef-98765432
NAME:The projector
RESOURCE-TYPE:projector
STRUCTURED-DATA;VALUE=URI:http://dir.example.com/projectors/3d.vcf
END:VRESOURCE
```

## 8. Extended examples

The following are some examples of the use of the properties defined in this specification. They include additional properties defined in [RFC7986] which includes IMAGE.

### 8.1. Example 1

The following is an example of a VEVENT describing a concert. It includes location information for the venue itself as well as references to parking and restaurants.

```
BEGIN:VEVENT
CREATED:20200215T145739Z
DESCRIPTION: Piano Sonata No 3\n
    Piano Sonata No 30
DTSTAMP:20200215T145739Z
DTSTART;TZID=America/New_York:20200315T150000Z
DTEND;TZID=America/New_York:20200315T163000Z
LAST-MODIFIED:20200216T145739Z
SUMMARY:Beethoven Piano Sonatas
UID:123456
IMAGE;VALUE=URI;DISPLAY=BADGE;FMTPROPERTY=image/png:h
    ttp://example.com/images/concert.png
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:SPONSOR
UID:dG9tQGZvb2Jhci5x1LmNvbQ
STRUCTURED-DATA;VALUE=URI:http://example.com/sponsor.vcf
END:PARTICIPANT
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:PERFORMER:
UID:em9lQGZvb2GFtcGx1LmNvbQ
STRUCTURED-DATA;VALUE=URI:http://www.example.com/people/johndoe.vcf
END:PARTICIPANT
BEGIN:VLOCATION
UID:123456-abcdef-98765432
NAME:The venue
STRUCTURED-DATA;VALUE=URI:http://dir.example.com/venues/big-hall.vcf
END:VLOCATION
BEGIN:VLOCATION
UID:123456-abcdef-87654321
NAME:Parking for the venue
STRUCTURED-DATA;VALUE=URI:http://dir.example.com/venues/parking.vcf
END:VLOCATION
END:VEVENT
```

## 8.2. Example 2

The following is an example of a VEVENT describing a meeting. One of the attendees is a remote participant.

```
BEGIN:VEVENT
CREATED:20200215T145739Z
DTSTAMP:20200215T145739Z
DTSTART;TZID=America/New_York:20200315T150000Z
DTEND;TZID=America/New_York:20200315T163000Z
LAST-MODIFIED:20200216T145739Z
SUMMARY:Conference planning
UID:123456
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=ACCEPTED;CN=A:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CN=B:mailto:b@example.com
BEGIN:PARTICIPANT
PARTICIPANT-TYPE:ACTIVE:
UID:v39lQGZvb2GFtcGx1LmNvbQ
STRUCTURED-DATA;VALUE=URI:http://www.example.com/people/b.vcf
LOCATION:At home
END:PARTICIPANT
END:VEVENT
```

## 9. Security Considerations

This specification extends [RFC5545] and makes further use of possibly linked data. While calendar data is not unique in this regard it is worth reminding implementors of some of the dangers and safeguards.

### 9.1. URIs

See [RFC3986] for a discussion of the security considerations relating to URIs. Because of the issues discussed there and below, clients SHOULD NOT follow URIs and fetch content automatically, and should only do so at the explicit request of the user.

Fetching remote resources carries inherent risks. Connections must only be allowed on well known ports, using allowed protocols (generally just HTTP/HTTPS on their default ports). The URL must be resolved externally and not allowed to access internal resources. Connecting to an external source reveals IP (and therefore generally location) information.

A maliciously constructed iCalendar object may contain a very large number of URIs. In the case of published calendars with a large

number of subscribers, such objects could be widely distributed. Implementations should be careful to limit the automatic fetching of linked resources to reduce the risk of this being an amplification vector for a denial-of-service attack.

## 9.2. Malicious Content

For the "STRUCTURED-DATA" property, agents need to be aware that a client could attack underlying storage by sending extremely large values and could attack processing time by uploading a recurring event with a large number of overrides and then repeatedly adding, updating and deleting structured data.

Agents should set reasonable limits on storage size and number of instances and apply those constraints. Calendar protocols should ensure there is a way to report on such limits being exceeded.

Malicious content could be introduced into the calendar server by way of the "STRUCTURED-DATA" property and propagated to many end users via scheduling. Servers SHOULD check this property for malicious or inappropriate content. Upon detecting such content, servers SHOULD remove the property,

## 9.3. HTML Content

When processing HTML content, applications need to be aware of the many security and privacy issues, as described in the IANA considerations section of [W3C.REC-html51-20171003]

## 10. Privacy Considerations

### 10.1. Tracking

Properties with a "URI" value type can expose their users to privacy leaks as any network access of the URI data can be tracked both by a network observer and by the entity hosting the remote resource. Clients SHOULD NOT automatically download data referenced by the URI without explicit instruction from users.

To help alleviate some of the concerns protocols and services could provide proxy services for downloading referenced data.

### 10.2. Revealing Locations

The addition of location information to the new participant component provides information about the location of participants at a given time. This information MUST NOT be distributed to other participants without those participant's express permission. Note that there may

be a number of participants who may be unaware of their inclusion in the data.

Agents processing and distributing calendar data must be aware that it has the property of providing information about a future time when a given individual may be at a particular location, which could enable targeted attacks against that individual.

The same may be true of other information contained in the participant component. In general, revealing only as much as is absolutely necessary should be the approach taken.

For example, there may be some privacy considerations relating to the ORDER parameter, as it provides an indication of the organizer's perception of the relative importance of other participants.

## 11. IANA Considerations

### 11.1. Additional iCalendar Registrations

#### 11.1.1. Properties

This document defines the following new iCalendar properties to be added to the registry defined in Section 8.2.3 of [RFC5545]:

Property	Status	Reference
CALENDAR-ADDRESS	Current	RFCXXXX, Section 6.4
LOCATION-TYPE	Current	RFCXXXX, Section 6.1
PARTICIPANT-TYPE	Current	RFCXXXX, Section 6.2
RESOURCE-TYPE	Current	RFCXXXX, Section 6.3
STRUCTURED-DATA	Current	RFCXXXX, Section 6.6
STYLED-DESCRIPTION	Current	RFCXXXX, Section 6.5

#### 11.1.2. Parameters

This document defines the following new iCalendar property parameters to be added to the registry defined in Section 8.2.4 of [RFC5545]:

Property Parameter	Status	Reference
ORDER	Current	RFCXXXX, Section 5.1
SCHEMA	Current	RFCXXXX, Section 5.2
DERIVED	Current	RFCXXXX, Section 5.3

### 11.1.3. Components

This document defines the following new iCalendar components to be added to the registry defined in Section 8.3.1 of [RFC5545]:

Component	Status	Reference
PARTICIPANT	Current	RFCXXXX, Section 7.1
VLOCATION	Current	RFCXXXX, Section 7.2
VRESOURCE	Current	RFCXXXX, Section 7.3

### 11.2. New Registration Tables

This section defines new registration tables for PARTICIPANT-TYPE and RESOURCE-TYPE values. These tables are updated using the same approaches laid down in Section 8.2.1 of [RFC5545]

This document creates new IANA registries for participant and resource types. IANA will maintain these registries and, following the policies outlined in [RFC8126], new tokens are assigned after Expert Review. The Expert Reviewer will generally consult the IETF GeoPRIV working group mailing list or its designated successor. Updates or deletions of tokens from the registration follow the same procedures. The expert review should be guided by a few common sense considerations. For example, tokens should not be specific to a country, region, organization, or company; they should be well-defined and widely recognized. The expert's support of IANA will include providing IANA with the new token(s) when the update is provided only in the form of a schema, and providing IANA with the new schema element(s) when the update is provided only in the form of a token. To ensure widespread usability across protocols, tokens MUST follow the character set restrictions for XML Names [3]. Each registration must include the name of the token and a brief description similar to the ones offered herein for the initial registrations contained this document:

#### 11.2.1. Participant Types

The following table has been used to initialize the participant types registry.



Participant Type	Status	Reference
ACTIVE	Current	RFCXXXX, Section 6.2
INACTIVE	Current	RFCXXXX, Section 6.2
SPONSOR	Current	RFCXXXX, Section 6.2
CONTACT	Current	RFCXXXX, Section 6.2
BOOKING-CONTACT	Current	RFCXXXX, Section 6.2
EMERGENCY-CONTACT	Current	RFCXXXX, Section 6.2
PUBLICITY-CONTACT	Current	RFCXXXX, Section 6.2
PLANNER-CONTACT	Current	RFCXXXX, Section 6.2
PERFORMER	Current	RFCXXXX, Section 6.2
SPEAKER	Current	RFCXXXX, Section 6.2

### 11.2.2. Resource Types

The following table has been used to initialize the resource types registry.

Resource Type	Status	Reference
PROJECTOR	Current	RFCXXXX, Section 6.3
ROOM	Current	RFCXXXX, Section 6.3
REMOTE-CONFERENCE-AUDIO	Current	RFCXXXX, Section 6.3
REMOTE-CONFERENCE-VIDEO	Current	RFCXXXX, Section 6.3

## 12. Acknowledgements

The author would like to thank Chuck Norris of eventful.com for his work which led to the development of this RFC.

The author would also like to thank the members of CalConnect, The Calendaring and Scheduling Consortium, the Event Publication technical committee and the following individuals for contributing their ideas and support:

Cyrus Daboo, John Haug, Dan Mendell, Ken Murchison, Scott Otis.

## 13. Normative References

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

- [RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, DOI 10.17487/RFC2426, September 1998, <<https://www.rfc-editor.org/info/rfc2426>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4589] Schulzrinne, H. and H. Tschofenig, "Location Types Registry", RFC 4589, DOI 10.17487/RFC4589, July 2006, <<https://www.rfc-editor.org/info/rfc4589>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.
- [RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <<https://www.rfc-editor.org/info/rfc7986>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [W3C.REC-html51-20171003] Faulkner, S., Eicholz, A., Leithead, T., and A. Danilo, "HTML 5.1 2nd Edition", World Wide Web Consortium Recommendation REC-html51-20171003, October 2017, <<https://www.w3.org/TR/2017/REC-html51-20171003>>.

[W3C.REC-xml-20081126]

Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and  
F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth  
Edition)", World Wide Web Consortium Recommendation REC-  
xml-20081126, November 2008,  
<<https://www.w3.org/TR/2008/REC-xml-20081126>>.

#### Appendix A. Open issues

None at the moment

#### Appendix B. Change log

To be deleted on publication

calext-v19 2021-03-25 MD

- o Revert ABNF to RFC5545 format.
- o Add missing DERIVED parameter registration.
- o Fix small error in an example (missing space at start).

calext-v18 2021-??-?? MD

- o Fix incorrect participant type property name in PARTICIPANT.
- o Allow parameters on LOCATION-TYPE.

calext-v17 2021-01-03 MD

- o Remove STRUCTURED-LOCATION property, add VLOCATION component.
- o Remove STRUCTURED-RESOURCE property, add VRESOURCE component.
- o Make LOCATION-TYPE multi-valued property for location.
- o Make RESOURCE-TYPE multi-valued property for resource.
- o Tidy up abnf.

calext-v16 2019-10-09 MD

- o Make LOCTYPE multi-valued.
- o Add all ATTENDEE scheduling parameters to CALENDAR-ADDRESS.

calext-v15 2019-10-08 MD

- o Address various DICUSS points.

calext-v14 2019-06-11 MD

- o Definition of event and social calendaring.
- o Remove redefinition of SOURCE - use STRUCTURED-DATA.

calext-v13 2019-05-26 MD

- o Respond to various issues.

calext-v12 2019-02-28 MD

- o Fix styled-description example. Respond to various AD issues. Some typos.

calext-v11 2019-02-27 MD

- o Add DERIVED parameter for styled-description, RELATED parameter for structured-location

calext-v09 2018-08-30 MD

- o Sorted out inconsistencies in refs to 5546

calext-v08 2018-07-06 MD

- o Add some text for equal ORDER values
- o Switched scheduleaddress to calendaraddress in participant abnf. Also added more properties
- o Fixed PARTICIPANT abnf

calext-v04 2017-10-11 MD

- o Change SCHEDULE-ADDRESS to CALENDAR-ADDRESS
- o Explicitly broaden scope of SOURCE
- o Add initial registry for RESTYPE and move new tables into separate section.
- o Fix PARTTYPE/PARTICIPANT-TYPE inconsistency

calext-v03 2017-10-09 MD

- o Mostly typographical and other minor changes

calext-v02 2017-04-20 MD

- o Add SCHEDULE-ADDRESS property
- o PARTICIPANT becomes a component rather than a property. Turn many of the former parameters into properties.
- o Use existing ATTENDEE property for scheduling.

calext-v01 2017-02-18 MD

- o Change ASSOCIATE back to PARTICIPANT
- o PARTICIPANT becomes a component rather than a property. Turn many of the former parameters into properties.

calext-v00 2016-08-?? MD

- o Name changed - taken up by calext working group

v06 2016-06-26 MD

- o Fix up abnf
- o change ref to ietf from daboo
- o take out label spec - use Cyrus spec

v05 2016-06-14 MD

- o Remove GROUP and HASH. they can be dealt with elsewhere if desired
- o Change ORDER to integer  $\geq 1$ .
- o Incorporate Structured-Data into this specification.

v04 2014-02-01 MD

- o Added updates attribute.
- o Minor typos.
- o Resubmitted mostly to refresh the draft.

v03 2013-03-06 MD

- o Replace PARTICIPANT with ASSOCIATE plus related changes.
- o Added section showing modifications to components.
- o Replace ID with GROUP and modify HASH.
- o Replace TITLE param with LABEL.
- o Fixed STYLED-DESCRIPTION in various ways, correct example.

v02 2012-11-02 MD

- o Collapse sections with description of properties and the use cases into a section with sub-sections.
- o New section to describe relating properties.
- o Remove idref and upgrade hash to have the reference
- o No default value types on properties..

v01 2012-10-18 MD Many changes.

- o SPONSOR and STRUCTURED-CONTACT are now in PARTICIPANT
- o Add a STRUCTURED-RESOURCE property
- o STYLED-DESCRIPTION to handle rich text
- o Much more...

2011-01-07

- o Remove MEDIA - it's going in the Cyrus RFC
- o Rename EXTENDED-... to STRUCTURED-...
- o Add TYPE parameter to SPONSOR

v00 2007-10-19 MD Initial version

Author's Address

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

Email: [mdouglass@bedework.com](mailto:mdouglass@bedework.com)  
URI: <http://bedework.com>

Calendaring extensions  
Internet-Draft  
Intended status: Standards Track  
Expires: April 19, 2021

N. Jenkins  
R. Stepanek  
Fastmail  
October 16, 2020

JSCalendar: A JSON representation of calendar data  
draft-ietf-calext-jscalendar-32

## Abstract

This specification defines a data model and JSON representation of calendar data that can be used for storage and data exchange in a calendaring and scheduling environment. It aims to be an alternative and, over time, successor to the widely deployed iCalendar data format, and to be unambiguous, extendable, and simple to process. In contrast to the jCal format, which is also JSON-based, JSCalendar is not a direct mapping from iCalendar, but defines the data model independently 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 April 19, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must



include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1.	Introduction . . . . .	4
1.1.	Motivation and Relation to iCalendar and jCal . . . . .	5
1.2.	Notational Conventions . . . . .	6
1.3.	Type Signatures . . . . .	6
1.4.	Data Types . . . . .	7
1.4.1.	Id . . . . .	7
1.4.2.	Int . . . . .	7
1.4.3.	UnsignedInt . . . . .	8
1.4.4.	UTCDateTime . . . . .	8
1.4.5.	LocalDateTime . . . . .	8
1.4.6.	Duration . . . . .	9
1.4.7.	SignedDuration . . . . .	10
1.4.8.	TimeZoneId . . . . .	10
1.4.9.	PatchObject . . . . .	11
1.4.10.	Relation . . . . .	12
1.4.11.	Link . . . . .	12
2.	JSCalendar Objects . . . . .	14
2.1.	JSEvent . . . . .	14
2.2.	JSTask . . . . .	14
2.3.	JSGroup . . . . .	14
3.	Structure of JSCalendar Objects . . . . .	15
3.1.	Object Type . . . . .	15
3.2.	Normalization and Equivalence . . . . .	15
3.3.	Vendor-specific Property Extensions, Values and Types . . . . .	15
4.	Common JSCalendar Properties . . . . .	16
4.1.	Metadata Properties . . . . .	16
4.1.1.	@type . . . . .	16
4.1.2.	uid . . . . .	16
4.1.3.	relatedTo . . . . .	17
4.1.4.	prodId . . . . .	17
4.1.5.	created . . . . .	17
4.1.6.	updated . . . . .	18
4.1.7.	sequence . . . . .	18
4.1.8.	method . . . . .	18
4.2.	What and Where Properties . . . . .	18
4.2.1.	title . . . . .	18
4.2.2.	description . . . . .	18
4.2.3.	descriptionContentType . . . . .	18
4.2.4.	showWithoutTime . . . . .	19
4.2.5.	locations . . . . .	19
4.2.6.	virtualLocations . . . . .	20
4.2.7.	links . . . . .	21

4.2.8.	locale	21
4.2.9.	keywords	22
4.2.10.	categories	22
4.2.11.	color	22
4.3.	Recurrence Properties	22
4.3.1.	recurrenceId	23
4.3.2.	recurrenceRules	23
4.3.3.	excludedRecurrenceRules	31
4.3.4.	recurrenceOverrides	32
4.3.5.	excluded	33
4.4.	Sharing and Scheduling Properties	33
4.4.1.	priority	33
4.4.2.	freeBusyStatus	33
4.4.3.	privacy	34
4.4.4.	replyTo	35
4.4.5.	participants	35
4.5.	Alerts Properties	41
4.5.1.	useDefaultAlerts	41
4.5.2.	alerts	41
4.6.	Multilingual Properties	43
4.6.1.	localizations	43
4.7.	Time Zone Properties	44
4.7.1.	timeZone	44
4.7.2.	timeZones	44
5.	Type-specific JSCalendar Properties	47
5.1.	JSEvent Properties	47
5.1.1.	start	47
5.1.2.	duration	47
5.1.3.	status	47
5.2.	JSTask Properties	48
5.2.1.	due	48
5.2.2.	start	48
5.2.3.	estimatedDuration	48
5.2.4.	percentComplete	48
5.2.5.	progress	48
5.2.6.	progressUpdated	49
5.3.	JSGroup Properties	49
5.3.1.	entries	50
5.3.2.	source	50
6.	Examples	50
6.1.	Simple event	51
6.2.	Simple task	51
6.3.	Simple group	51
6.4.	All-day event	52
6.5.	Task with a due date	52
6.6.	Event with end time zone	53
6.7.	Floating-time event (with recurrence)	53
6.8.	Event with multiple locations and localization	54

6.9.	Recurring event with overrides . . . . .	55
6.10.	Recurring event with participants . . . . .	56
7.	Security Considerations . . . . .	58
7.1.	Expanding Recurrences . . . . .	58
7.2.	JSON Parsing . . . . .	59
7.3.	URI Values . . . . .	59
7.4.	Spam . . . . .	60
7.5.	Duplication . . . . .	60
7.6.	Time Zones . . . . .	60
8.	IANA Considerations . . . . .	61
8.1.	Media Type Registration . . . . .	61
8.2.	Creation of "JSCalendar Properties" Registry . . . . .	62
8.2.1.	Preliminary Community Review . . . . .	63
8.2.2.	Submit Request to IANA . . . . .	63
8.2.3.	Designated Expert Review . . . . .	63
8.2.4.	Change Procedures . . . . .	63
8.2.5.	JSCalendar Properties Registry Template . . . . .	64
8.2.6.	Initial Contents for the JSCalendar Properties Registry . . . . .	64
8.3.	Creation of "JSCalendar Types" Registry . . . . .	73
8.3.1.	JSCalendar Types Registry Template . . . . .	73
8.3.2.	Initial Contents for the JSCalendar Types Registry . . . . .	73
8.4.	Creation of "JSCalendar Enum Values" Registry . . . . .	74
8.4.1.	JSCalendar Enum Property Template . . . . .	74
8.4.2.	JSCalendar Enum Value Template . . . . .	75
8.4.3.	Initial Contents for the JSCalendar Enum Values registry . . . . .	75
9.	Acknowledgments . . . . .	81
10.	References . . . . .	81
10.1.	Normative References . . . . .	82
10.2.	Informative References . . . . .	84
	Authors' Addresses . . . . .	85

## 1. Introduction

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

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

- o The attributes of the calendar entry represented must be described as simple key-value pairs. Simple events are simple to represent; complex events can be modelled accurately.
- o Wherever possible, there should be only one way to express the desired semantics, reducing complexity.

- o The data model should avoid ambiguities, which often lead to interoperability issues between implementations.
- o The data model should be generally compatible with the iCalendar data format [RFC5545] [RFC7986] and extensions, but the specification should add new attributes where the iCalendar format currently lacks expressivity, and drop seldom-used, obsolete, or redundant properties. This means translation with no loss of semantics should be easy with most common iCalendar files.
- o Extensions, such as new properties and components, should not require updates to this document.

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

#### 1.1. Motivation and Relation to iCalendar and jCal

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

Sources of implementation errors include the following:

- o iCalendar defines various formats for local times, UTC time, and dates.
- o iCalendar requires custom time zone definitions within a single calendar component.
- o iCalendar's definition of recurrence rules is ambiguous and has resulted in differing understandings even between experienced calendar developers.
- o The iCalendar format itself causes interoperability issues due to misuse of CRLF-terminated strings, line continuations, and subtle differences among iCalendar parsers.

In recent years, many new products and services have appeared that wish to use a JSON representation of calendar data within their APIs. The JSON format for iCalendar data, jCal [RFC7265], is a direct mapping between iCalendar and JSON. In its effort to represent full

iCalendar semantics, it inherits all the same pitfalls and uses a complicated JSON structure.

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

## 1.2. Notational Conventions

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

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

## 1.3. Type Signatures

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

- o "\*" - The type is undefined (the value could be any type, although permitted values may be constrained by the context of this value).
- o "String" - The JSON string type.
- o "Number" - The JSON number type.
- o "Boolean" - The JSON boolean type.
- o "A[B]" - A JSON object where the keys are all of type "A", and the values are all of type "B".

- o "A[]" - An array of values of type "A".
- o "A|B" - The value is either of type "A" or of type "B".

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

#### 1.4. Data Types

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

##### 1.4.1. 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 alphanumeric characters ("A-Za-z0-9"), hyphen ("-"), and underscore ("\_").

In many places in JSCalendar 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 JSCalendar 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 JSEvent objects might use the same Ids in their respective "links" properties. Or within the same JSEvent object the same Id could appear in the "participants" and "alerts" properties. These situations do not imply any semantic connections among the objects.

Nevertheless, a UUID [RFC4122] is typically a good choice.

##### 1.4.2. Int

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

#### 1.4.3. UnsignedInt

Where "UnsignedInt" is given as a data type, it means an integer in the range  $0 \leq \text{value} \leq 2^{53}-1$ , represented as a JSON "Number".

#### 1.4.4. 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 "2010-10-10T10:10:10Z".

#### 1.4.5. LocalDateTime

This is a date-time string with no time zone/offset information. It is otherwise in the same format as UTCDateTime, including fractional seconds. For example "2006-01-02T15:04:05" and "2006-01-02T15:04:05.003" are both valid. The time zone to associate with the LocalDateTime comes from the "timeZone" property of the JSCalendar object (see Section 4.7.1). If no time zone is specified, the LocalDateTime is "floating". Floating date-times are not tied to any specific time zone. Instead, they occur in each time zone at the given wall-clock time (as opposed to the same instant point in time).

A time zone may have a period of discontinuity, for example a change from standard time to daylight-savings time. When converting local date-times that fall in the discontinuity to UTC, the offset before the transition MUST be used.

For example, in the America/Los\_Angeles time zone, the date-time 2020-11-01T01:30:00 occurs twice: before the DST transition with a UTC offset of -07:00, and again after the transition with an offset of -08:00. When converting to UTC, we therefore use the offset before the transition (-07:00) and so it becomes 2020-11-01T08:30:00Z.

Similarly, in the Australia/Melbourne time zone, the date-time 2020-10-04T02:30:00 does not exist: the clocks are moved forward one hour for DST on that day at 02:00. However, such a value may appear during calculations (see duration semantics in Section 1.4.6), or due to a change in time zone rules (so it was valid when the event was first created). Again, it is interpreted as though the offset before

the transition is in effect (+10:00), therefore when converted to UTC we get 2020-10-03T16:30:00Z.

#### 1.4.6. Duration

Where Duration is given as a type, it means a length of time represented by a subset of ISO8601 duration format, as specified by the following ABNF [RFC5234]:

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

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

In addition, the duration MUST NOT include fractional second values unless the fraction is non-zero. Fractional second values MUST NOT have trailing zeros, to ensure there is only a single representation for each duration.

A duration specifies an abstract number of weeks, days, hours, minutes, and/or seconds. A duration specified using weeks or days does not always correspond to an exact multiple of 24 hours. The number of hours/minutes/seconds may vary if it overlaps a period of discontinuity in the event's time zone, for example a change from standard time to daylight-savings time. Leap seconds MUST NOT be considered when adding or subtracting a duration to/from a `LocalDateTime`.

To add a duration to a `LocalDateTime`:

1. Add any week or day components of the duration to the date. A week is always the same as 7 days.
2. If a time zone applies to the `LocalDateTime`, convert it to a `UTCDateTime` following the semantics in Section 1.4.5.
3. Add any hour, minute or second components of the duration (in absolute time).
4. Convert the resulting `UTCDateTime` back to a `LocalDateTime` in the time zone that applies.



To subtract a duration from a `LocalDateTime`, the steps apply in reverse:

1. If a time zone applies to the `LocalDateTime`, convert it to UTC following the semantics in Section 1.4.5.
2. Subtract any hour, minute or second components of the duration (in absolute time).
3. Convert the resulting `UTCDateTime` back to `LocalDateTime` in the time zone that applies.
4. Subtract any week or day components of the duration from the date.
5. If the resulting time does not exist on the date due to a discontinuity in the time zone, use the semantics in Section 1.4.5 to convert to UTC and back to get a valid `LocalDateTime`.

These semantics match the iCalendar DURATION value type ([RFC5545], Section 3.3.6).

#### 1.4.7. SignedDuration

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

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

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

#### 1.4.8. TimeZoneId

Where "TimeZoneId" is given as a data type, it means a "String" that is either a time zone name in the IANA Time Zone Database [TZDB] or a custom time zone identifier defined in the "timeZones" property (see Section 4.7.2).

Where an IANA time zone is specified, the zone rules of the respective zone records apply. Custom time zones are interpreted as described in Section 4.7.2.

#### 1.4.9. 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., "alerts/1/offset" and "alerts".
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:

- o If null, remove the property from the patched object. If the key is not present in the parent, this a no-op.
- o Anything else: The value to set for this property (this may be a replacement or addition to the object being patched).

A PatchObject does not define its own "@type" property (see Section 4.1.1). 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.

The PatchObject format is used to significantly reduce file size and duplicated content when specifying variations to a common object, such as with recurring events or when translating the data into multiple languages. It can also better preserve semantic intent if only the properties that should differ between the two objects are

patched. For example, if one person is not going to a particular instance of a regularly scheduled event, in iCalendar you would have to duplicate the entire event in the override. In JSCalendar this is a small patch to show the difference. As only this property is patched, if the location of the event is changed, the occurrence will automatically still inherit this.

#### 1.4.10. Relation

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

- o @type: "String" (mandatory)

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

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

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

Keys in the set MUST be one of the following values, or specified in the property definition where the Relation object is used, or a value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- \* "first": The linked object is the first in a series the linking object is part of.
- \* "next": The linked object is the next in a series the linking object is part of.
- \* "child": The linked object is a subpart of the linking object.
- \* "parent": The linking object is a subpart of the linked object.

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

#### 1.4.11. Link

A Link object represents an external resource associated with the linking object. It has the following properties:

- o @type: "String" (mandatory)

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

- o href: "String" (mandatory)

A URI [RFC3986] from which the resource may be fetched.

This MAY be a "data:" URL [RFC2397], but it is recommended that the file be hosted on a server to avoid embedding arbitrarily large data in JSCalendar object instances.

- o cid: "String" (optional)

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

- o contentType: "String" (optional)

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

- o size: "UnsignedInt" (optional)

The size, in octets, of the resource when fully decoded (i.e., the number of octets in the file the user would download), if known. Note that this is an informational estimate, and implementations must be prepared to handle the actual size being quite different when the resource is fetched.

- o rel: "String" (optional)

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

- o display: "String" (optional)

Describes the intended purpose of a link to an image. If set, the "rel" property MUST be set to "icon". The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- \* "badge": an image meant to be displayed alongside the title of the object.

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

- \* "fullsize": an image that is used to enhance the object.
- \* "thumbnail": a smaller variant of "fullsize" to be used when space for the image is constrained.
- o title: "String" (optional)
  - A human-readable plain-text description of the resource.

## 2. JSCalendar Objects

This section describes the calendar object types specified by JSCalendar.

### 2.1. JSEvent

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

A JSEvent represents a scheduled amount of time on a calendar, typically a meeting, appointment, reminder or anniversary. It is required to start at a certain point in time and typically has a non-zero duration. Multiple participants may partake in the event at multiple locations.

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

### 2.2. JSTask

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

A JSTask represents an action-item, assignment, to-do or work item. It may start and be due at certain points in time, may take some estimated time to complete, and may recur, none of which is required.

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

### 2.3. JSGroup

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

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

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

### 3. Structure of JSCalendar Objects

A JSCalendar object is a JSON object [RFC8259], which MUST be valid I-JSON (a stricter subset of JSON) [RFC7493]. Property names and values are case-sensitive.

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

#### 3.1. Object Type

JSCalendar objects MUST name their type in the "@type" property, if not explicitly specified otherwise for the respective object type. A notable exception to this rule is the PatchObject (Section 1.4.9).

#### 3.2. Normalization and Equivalence

JSCalendar aims to provide unambiguous definitions for value types and properties, but does not define a general normalization or equivalence method for JSCalendar objects and types. This is because the notion of equivalence might range from byte-level equivalence to semantic equivalence, depending on the respective use case.

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

- o Custom JSCalendar properties may contain arbitrary JSON values, including arrays. However, equivalence of arrays might or might not depend on the order of elements, depending on the respective property definition.
- o Several JSCalendar property values are defined as URIs and media types, but normalization of these types is inherently protocol- and scheme-specific, depending on the use-case of the equivalence definition (see Section 6 of [RFC3986]).

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

#### 3.3. Vendor-specific Property Extensions, Values and Types

Vendors MAY add additional properties to the calendar 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 JSCalendar object, it either MUST include a "@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 JSCalendar 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.

#### 4. Common JSCalendar Properties

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

##### 4.1. Metadata Properties

###### 4.1.1. @type

Type: "String" (mandatory).

Specifies the type which this object represents. This MUST be one of the following values:

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

###### 4.1.2. uid

Type: "String" (mandatory).

A globally unique identifier, used to associate the object as the same across different systems, calendars and views. The value of this property MUST be unique across all JSCalendar objects, even if they are of different type. [RFC4122] describes a range of established algorithms to generate universally unique identifiers (UUID). UUID version 4, described in Section 4.4 of [RFC4122], is RECOMMENDED.

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

#### 4.1.3. relatedTo

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

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

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

#### 4.1.4. prodId

Type: "String" (optional).

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

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

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

#### 4.1.5. created

Type: "UTCDateTime" (optional).

The date and time this object was initially created.



#### 4.1.6. updated

Type: "UTCDateTime" (mandatory).

The date and time the data in this object was last modified (or its creation date/time if not modified since).

#### 4.1.7. sequence

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

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

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

#### 4.1.8. method

Type: "String" (optional).

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

### 4.2. What and Where Properties

#### 4.2.1. title

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

A short summary of the object.

#### 4.2.2. description

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

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

#### 4.2.3. descriptionContentType

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

Describes the media type [RFC6838] of the contents of the "description" property. Media types MUST be sub-types of type "text", and SHOULD be "text/plain" or "text/html" [MEDIATYPES]. They MAY include parameters and the "charset" parameter value MUST be

"utf-8", if specified. Descriptions of type "text/html" MAY contain "cid" URLs [RFC2392] to reference links in the calendar object by use of the "cid" property of the Link object.

#### 4.2.4. showWithoutTime

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

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

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

#### 4.2.5. locations

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

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

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

- o @type: "String" (mandatory)

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

- o name: "String" (optional)

The human-readable name of the location.

- o description: "String" (optional)

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

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

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

the keys being the location types. The value for each key in the map MUST be true.

- o `relativeTo: "String"` (optional)

Specifies the relation between this location and the time of the JSCalendar object. This is primarily to allow events representing travel to specify the location of departure (at the start of the event) and location of arrival (at the end); this is particularly important if these locations are in different time zones, as a client may wish to highlight this information for the user.

This MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3). Any value the client or server doesn't understand should be treated the same as if this property is omitted.

- \* `"start"`: The event/task described by this JSCalendar object occurs at this location at the time the event/task starts.

- \* `"end"`: The event/task described by this JSCalendar object occurs at this location at the time the event/task ends.

- o `timeZone: "TimeZoneId"` (optional)

A time zone for this location.

- o `coordinates: "String"` (optional)

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

- o `links: "Id[Link]"` (optional)

A map of link ids to Link objects, representing external resources associated with this location, for example a vCard or image. If there are no links, this MUST be omitted (rather than specified as an empty set).

#### 4.2.6. `virtualLocations`

Type: `"Id[VirtualLocation]"` (optional).

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

A VirtualLocation object has the following properties.

- o @type: "String" (mandatory)

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

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

The human-readable name of the virtual location.

- o description: "String" (optional)

Human-readable plain-text instructions for accessing this virtual location. This may be a conference access code, etc.

- o uri: "String" (mandatory)

A URI [RFC3986] that represents how to connect to this virtual location.

This may be a telephone number (represented using the "tel:" scheme, e.g., "tel:+1-555-555-5555") for a teleconference, a web address for online chat, or any custom URI.

#### 4.2.7. links

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

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

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

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

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

#### 4.2.8. locale

Type: "String" (optional).

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

#### 4.2.9. keywords

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

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

#### 4.2.10. categories

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

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

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

#### 4.2.11. color

Type: "String" (optional).

A color clients MAY use when displaying this calendar object. The value is a color name taken from the set of names defined in Section 4.3 of CSS Color Module Level 3 [COLORS], or an RGB value in hexadecimal notation, as defined in Section 4.2.1 of CSS Color Module Level 3.

### 4.3. Recurrence Properties

Some events and tasks occur at regular or irregular intervals. Rather than having to copy the data for every occurrence there can be a master event with rules to generate recurrences, and/or overrides that add extra dates or exceptions to the rules.

The recurrence set is the complete set of instances for an object. It is generated by considering the following properties in order, all of which are optional:

1. The recurrenceRules property (Section 4.3.2) generates a set of extra date-times on which the object occurs.
2. The excludedRecurrenceRules property (Section 4.3.3) generates a set of date-times that are to be removed from the previously generated set of date-times on which the object occurs.

3. The `recurrenceOverrides` property (Section 4.3.4) defines date-times which are added or excluded to form the final set. (This property may also contain changes to the object to apply to particular instances.)

#### 4.3.1. `recurrenceId`

Type: `"LocalDateTime"` (optional).

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

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

#### 4.3.2. `recurrenceRules`

Type: `"RecurrenceRule[]"` (optional).

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

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

A JSTask recurs by applying the recurrence rules to the `"start"` date-time, if defined, otherwise it recurs by the `"due"` date-time, if defined. If the task defines neither a `"start"` nor `"due"` date-time, it MUST NOT define a `"recurrenceRules"` property.

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

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

- o `@type: "String"` (mandatory)

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

- o `frequency: "String"` (mandatory)

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

- \* "yearly"
- \* "monthly"
- \* "weekly"
- \* "daily"
- \* "hourly"
- \* "minutely"
- \* "secondly"

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

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

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

This is the INTERVAL part from iCalendar.

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

The calendar system in which this recurrence rule operates, in lowercase. This MUST be either a CLDR-registered calendar system name [CLDR], or a vendor-specific value (see Section 3.3).

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

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

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

- \* "omit"
- \* "backward"
- \* "forward"

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

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

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

- \* "mo"
- \* "tu"
- \* "we"
- \* "th"
- \* "fr"
- \* "sa"
- \* "su"

This is the WKST part from iCalendar.

- o byDay: "NDay[]" (optional)

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

- \* @type: "String" (mandatory)

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

- \* day: "String" (mandatory)

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

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

- \* nthOfPeriod: "Int" (optional)

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

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



- o byMonthDay: "Int[]" (optional)

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

This is the BYMONTHDAY part in iCalendar.

- o byMonth: "String[]" (optional)

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

This is the BYMONTH part from iCalendar.

- o byYearDay: "Int[]" (optional)

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

This is the BYYEARDAY part from iCalendar.

- o byWeekNo: "Int[]" (optional)

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

This is the BYWEEKNO part from iCalendar.

- o byHour: "UnsignedInt[]" (optional)

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

- o byMinute: "UnsignedInt[]" (optional)

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

This is the BYMINUTE part from iCalendar.

- o bySecond: "UnsignedInt[]" (optional)

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

This is the BYSECOND part from iCalendar.

- o bySetPosition: "Int[]" (optional)

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

- o count: "UnsignedInt" (optional)

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

This is the COUNT part from iCalendar.

- o until: "LocalDateTime" (optional)

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

This is the UNTIL part from iCalendar.

#### 4.3.2.1. Interpreting recurrence rules

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

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

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

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

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

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

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

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

- \* "weekly": every second from midnight (inclusive) on the first day of the week (as defined by the firstDayOfWeek property, or Monday if omitted), to midnight 7 days later (exclusive).
- \* "daily": every second from midnight at the start of the day (inclusive) to midnight at the end of the day (exclusive).
- \* "hourly": every second from the beginning of the hour (inclusive) to the beginning of the next hour (exclusive).
- \* "minutely": every second from the beginning of the minute (inclusive) to the beginning of the next minute (exclusive).
- \* "secondly": the second itself, only.

2. Each date-time candidate is compared against all of the byX properties of the rule except bySetPosition. If any property in the rule does not match the date-time, the date-time is eliminated. Each byX property is an array; the date-time matches

the property if it matches any of the values in the array. The properties have the following semantics:

- \* `byMonth`: the date-time is in the given month.
- \* `byWeekNo`: the date-time is in the nth week of the year. Negative numbers mean the nth last week of the year. This corresponds to weeks according to week numbering as defined in ISO.8601.2004, with a week defined as a seven day period, starting on the `firstDayOfWeek` property value or Monday if omitted. Week number one of the calendar year is the first week that contains at least four days in that calendar year.

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

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

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

- \* `byMonthDay`: the date-time is on the given day of the month. Negative numbers mean the nth last day of the month.
- \* `byDay`: the date-time is on the given day of the week. If the day is prefixed by a number, it is the nth occurrence of that day of the week within the month (if frequency is monthly) or year (if frequency is yearly). Negative numbers means nth last occurrence within that period.
- \* `byHour`: the date-time has the given hour value.
- \* `byMinute`: the date-time has the given minute value.
- \* `bySecond`: the date-time has the given second value.

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

1. After applying the `byMonth` filter, if the candidate's month is invalid for the given year, increment it (if `skip` is "forward") or decrement it (if `skip` is "backward") until a valid month is found, incrementing/decrementing the year as

well if passing through the beginning/end of the year. This only applies to calendar systems with leap months.

2. After applying the `byMonthDay` filter, if the day of the month is invalid for the given month and year, change the date to the first day of the next month (if `skip` is "forward") or the last day of the current month (if `skip` is "backward").
3. If any valid date produced after applying the `skip` is already a candidate, eliminate the duplicate. (For example after adjusting, 30th February and 31st February would both become the same "real" date, so one is eliminated as a duplicate.)
3. If a `bySetPosition` property is included, this is now applied to the ordered list of remaining dates. This property specifies the indexes of date-times to keep; all others should be eliminated. Negative numbers are indexes from the end of the list, with -1 being the last item.
4. Any date-times before the start date of the event are eliminated (see below for why this might be needed).
5. If a `skip` property is included and is not "omit", eliminate any date-times that have already been produced by previous iterations of the algorithm. (This is not possible if `skip` is "omit".)
6. If further dates are required (we have not reached the until date, or count limit) skip the next (interval - 1) sets of candidates, then continue from step 1.

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

1. The initial date-time to which the rule is applied (the "start" date-time for events; the "start" or "due" date-time for tasks) is always the first occurrence in the expansion (and is counted if the recurrence is limited by a "count" property), even if it would normally not match the rule.
2. The first set of candidates to consider is that which would contain the initial date-time. This means the first set may include candidates before the initial date-time; such candidates are eliminated from the results in step (4) as outlined before.
3. The following properties MUST be implicitly added to the rule under the given conditions:

- \* If frequency is not "secondly" and no bySecond property: Add a bySecond property with the sole value being the seconds value of the initial date-time.
- \* If frequency is not "secondly" or "minutely", and no byMinute property: Add a byMinute property with the sole value being the minutes value of the initial date-time.
- \* If frequency is not "secondly", "minutely" or "hourly" and no byHour property: Add a byHour property with the sole value being the hours value of the initial date-time.
- \* If frequency is "weekly" and no byDay property: Add a byDay property with the sole value being the day-of-the-week of the initial date-time.
- \* If frequency is "monthly" and no byDay property and no byMonthDay property: Add a byMonthDay property with the sole value being the day-of-the-month of the initial date-time.
- \* If frequency is "yearly" and no byYearDay property:
  - + If there are no byMonth or byWeekNo properties, and either there is a byMonthDay property or there is no byDay property: Add a byMonth property with the sole value being the month of the initial date-time.
  - + If there is no byMonthDay, byWeekNo or byDay properties: Add a byMonthDay property with the sole value being the day-of-the-month of the initial date-time.
  - + If there is a byWeekNo property and no byMonthDay or byDay properties: Add a byDay property with the sole value being the day-of-the-week of the initial date-time.

#### 4.3.3. excludedRecurrenceRules

Type: "RecurrenceRule[]" (optional).

Defines a set of recurrence rules (repeating patterns) for date-times on which the object will not occur. The rules are interpreted the same as for the "recurrenceRules" property (see Section 4.3.2), with the exception that the initial date-time to which the rule is applied (the "start" date-time for events; the "start" or "due" date-time for tasks) is only considered part of the expansion if it matches the rule. The resulting set of date-times are then removed from those generated by the recurrenceRules property, as described in Section 4.3.

#### 4.3.4. recurrenceOverrides

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

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

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

If the patch object defines the "excluded" property of an occurrence to be true, this occurrence is omitted from the final set of recurrences for the calendar object (like an EXDATE from iCalendar). Such a patch object MUST NOT patch any other property.

By default, an occurrence inherits all properties from the main object except the start (or due) date-time, which is shifted to match the recurrence id LocalDateTime. However, individual properties of the occurrence can be modified by a patch, or multiple patches. It is valid to patch the "start" property value, and this patch takes precedence over the value generated from the recurrence id. Both the recurrence id as well as the patched "start" date-time may occur before the original JSCalendar object's "start" or "due" date.

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

- o @type
- o excludedRecurrenceRules
- o method
- o privacy
- o prodId
- o recurrenceId
- o recurrenceOverrides
- o recurrenceRules
- o relatedTo

- o replyTo
- o uid

#### 4.3.5. excluded

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

Defines if this object is an overridden, excluded instance of a recurring JSCalendar object (see Section 4.3.4). If this property value is true, this calendar object instance MUST be removed from the occurrence expansion. The absence of this property, or the presence of its default value false, indicates that this instance MUST be included in the occurrence expansion.

### 4.4. Sharing and Scheduling Properties

#### 4.4.1. priority

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

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

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

#### 4.4.2. freeBusyStatus

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

Specifies how this calendar object should be treated when calculating free-busy state. This MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

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



#### 4.4.3. privacy

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

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

This property MUST NOT affect the information sent to scheduled participants; it is only interpreted by protocols that share the calendar objects belonging to one user with other users.

The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3). Any value the client or server doesn't understand should be preserved but treated as equivalent to "private".

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

- \* @type
- \* created
- \* due
- \* duration
- \* estimatedDuration
- \* freeBusyStatus
- \* privacy
- \* recurrenceOverrides. Only patches which apply to another permissible property are allowed to be shared.
- \* sequence
- \* showWithoutTime

- \* start
- \* timeZone
- \* timeZones
- \* uid
- \* updated
- o "secret": The object is hidden completely (as though it did not exist) when the calendar this object is in is shared.

#### 4.4.4. replyTo

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

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

The following methods are defined:

- o "imip": The organizer accepts an iMIP [RFC6047] response at this email address. The value MUST be a "mailto:" URI.
- o "web": Opening this URI in a web browser will provide the user with a page where they can submit a reply to the organizer. The value MUST be a URL using the "https:" scheme.
- o "other": The organizer is identified by this URI but the method for submitting the response is undefined.

#### 4.4.5. participants

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

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

If this property is set and any participant has a `sendTo` property, then the `"replyTo"` property of this calendar object **MUST** define at least one reply method.

A Participant object has the following properties:

- o `@type: "String"` (mandatory)

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

- o `name: "String"` (optional)

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

- o `email: "String"` (optional)

The email address for the participant.

- o `description: "String"` (optional).

A plain text description of this participant. For example, this may include more information about their role in the event or how best to contact them.

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

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

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

The following methods are defined:

- \* `"imip"`: The participant accepts an iMIP [RFC6047] request at this email address. The value **MUST** be a `"mailto:"` URI. It **MAY** be different from the value of the participant's `"email"` property.
- \* `"other"`: The participant is identified by this URI but the method for submitting the invitation is undefined.

- o `kind: "String"` (optional)

What kind of entity this participant is, if known.

This MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3). Any value the client or server doesn't understand should be treated the same as if this property is omitted.

- \* `"individual"`: a single person
- \* `"group"`: a collection of people invited as a whole
- \* `"location"`: a physical location that needs to be scheduled, e.g., a conference room
- \* `"resource"`: a non-human resource other than a location, such as a projector

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

A set of roles that this participant fulfills.

At least one role MUST be specified for the participant. The keys in the set MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- \* `"owner"`: The participant is an owner of the object. This signifies they have permission to make changes to it that affect the other participants. Non-owner participants may only change properties that just affect themselves (for example setting their own alerts or changing their rsvp status).
- \* `"attendee"`: The participant is expected to be present at the event.
- \* `"optional"`: The participant's involvement with the event is optional. This is expected to be primarily combined with the `"attendee"` role.
- \* `"informational"`: The participant is copied for informational reasons, and is not expected to attend.
- \* `"chair"`: The participant is in charge of the event/task when it occurs.

- \* "contact": The participant is someone that may be contacted for information about the event.

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

- o locationId: "String" (optional)

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

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

- o language: "String" (optional)

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

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

The participation status, if any, of this participant.

The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- \* "needs-action": No status yet set by the participant.
- \* "accepted": The invited participant will participate.
- \* "declined": The invited participant will not participate.
- \* "tentative": The invited participant may participate.
- \* "delegated": The invited participant has delegated their attendance to another participant, as specified in the delegatedTo property.

- o participationComment: "String" (optional)

A note from the participant to explain their participation status.

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

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

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

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

The value MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- \* "server": The calendar server will send the scheduling messages.
- \* "client": The calendar client will send the scheduling messages.
- \* "none": No scheduling messages are to be sent to this participant.

- o `scheduleForceSend`: "Boolean" (optional, default: false)

A client may set the property on a participant to true to request that the server send a scheduling message to the participant when it would not normally do so (e.g. if no significant change is made to the object or the `scheduleAgent` is set to client). The property MUST NOT be stored in the JSCalendar object on the server or appear in a scheduling message.

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

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

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

- o `scheduleStatus`: "String[]" (optional)

A list of status codes, as defined in Section 3.8.8.3 of [RFC5545], returned from the processing of the most recent scheduling message sent to this participant.

Servers MUST only add or change this property when they send a scheduling message to the participant. Clients SHOULD NOT change

or remove this property if it was provided by the server. Clients MAY add, change, or remove the property for participants where the client is handling the scheduling.

This property MUST NOT be included in scheduling messages.

- o `scheduleUpdated`: "UTCDateTime" (optional)

The timestamp for the most recent response from this participant.

This is the "updated" property of the last response when using iTIP. It can be compared to the "updated" property in future responses to detect and discard older responses delivered out of order.

- o `invitedBy`: "String" (optional)

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

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

A set of participant ids that this participant has delegated their participation to. Each key in the set MUST be the id of a participant. The value for each key in the map MUST be true. If there are no delegates, this MUST be omitted (rather than specified as an empty set).

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

A set of participant ids that this participant is acting as a delegate for. Each key in the set MUST be the id of a participant. The value for each key in the map MUST be true. If there are no delegators, this MUST be omitted (rather than specified as an empty set).

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

A set of group participants that were invited to this calendar object, which caused this participant to be invited due to their membership in the group(s). Each key in the set MUST be the id of a participant. The value for each key in the map MUST be true. If there are no groups, this MUST be omitted (rather than specified as an empty set).

- o `links`: "Id[Link]" (optional)

A map of link ids to Link objects, representing external resources associated with this participant, for example a vCard or image. If there are no links, this MUST be omitted (rather than specified as an empty set).

- o progress: "String" (optional; only allowed for participants of a JSTask). Represents the progress of the participant for this task. It MUST NOT be set if the "participationStatus" of this participant is any value other than "accepted". See Section 5.2.5 for allowed values and semantics.
- o progressUpdated: "UTCDateTime" (optional; only allowed for participants of a JSTask). Specifies the date-time the progress property was last set on this participant. See Section 5.2.6 for allowed values and semantics.
- o percentComplete: "UnsignedInt" (optional; only allowed for participants of a JSTask). Represents the percent completion of the participant for this task. The property value MUST be a positive integer between 0 and 100.

#### 4.5. Alerts Properties

##### 4.5.1. useDefaultAlerts

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

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

##### 4.5.2. alerts

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

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

An Alert Object has the following properties:

- o @type: "String" (mandatory)

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



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

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

An "OffsetTrigger" object has the following properties:

- \* @type: "String" (mandatory)

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

- \* offset: "SignedDuration" (mandatory).

Defines the offset at which to trigger the alert relative to the time property defined in the "relativeTo" property of the alert. Negative durations signify alerts before the time property, positive durations signify alerts after.

- \* relativeTo: "String" (optional, default: "start")

Specifies the time property that the alert offset is relative to. The value MUST be one of:

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

An "AbsoluteTrigger" object has the following properties:

- \* @type: "String" (mandatory)

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

- \* when: "UTCDateTime" (mandatory).

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

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

- o acknowledged: "UTCDateTime" (optional)

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

For a recurring calendar object, setting the "acknowledged" property MUST NOT add a new override to the "recurrenceOverrides" property. If the alert is not already overridden, the acknowledged property MUST be set on the alert in the master event/task.

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

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

Relates this alert to other alerts in the same JSCalendar object. If the user wishes to snooze an alert, the application MUST create an alert to trigger after snoozing. This new snooze alert MUST set a parent relation to the identifier of the original alert.

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

Describes how to alert the user.

The value MUST be at most one of the following values, a value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- \* "display": The alert should be displayed as appropriate for the current device and user context.
- \* "email": The alert should trigger an email sent out to the user, notifying about the alert. This action is typically only appropriate for server implementations.

#### 4.6. Multilingual Properties

##### 4.6.1. localizations

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

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

See the description of PatchObject (Section 1.4.9) for the structure of the PatchObject. The patches are applied to the top-level calendar object. In addition, the "locale" property of the patched object is set to the language tag. All pointers for patches MUST end with one of the following suffixes; any patch that does not follow this MUST be ignored unless otherwise specified in a future RFC:

- o title
- o description
- o name

A patch MUST NOT have the prefix "recurrenceOverrides"; any localization of the override MUST be a patch to the localizations property inside the override instead. For example, a patch to "locations/abcd1234/title" is permissible, but a patch to "uid" or "recurrenceOverrides/2020-01-05T14:00:00/title" is not.

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

#### 4.7. Time Zone Properties

##### 4.7.1. timeZone

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

Identifies the time zone the object is scheduled in, or null for floating time. This is either a name from the IANA Time Zone Database [TZDB] or the TimeZoneId of a custom time zone from the "timeZones" property (Section 4.7.2). If omitted, this MUST be presumed to be null (i.e., floating time).

##### 4.7.2. timeZones

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

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

- o To avoid conflict with names in the IANA Time Zone Database [TZDB], it MUST start with the "/" character.
- o It MUST be a valid "paramtext" value as specified in Section 3.1. of [RFC5545].
- o At least one other property in the same JSCalendar object MUST reference a time zone using this identifier (i.e., orphaned time zones are not allowed).

An identifier need only be unique to this JSCalendar object. A JSCalendar object may be part in a hierarchy of other JSCalendar objects (say, a JSEvent is an entry in a JSGroup). In this case, the set of time zones is the sum of the time zone definitions of this object and its parent objects. If multiple time zones with the same identifier exist, then the definition closest to the calendar object in relation to its parents MUST be used. (In context of JSEvent, a time zone definition in its timeZones property has precedence over a definition of the same id in the JSGroup). Time zone definitions in any children of the calendar object MUST be ignored.

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

- o @type: "String" (mandatory)  
Specifies the type of this object. This MUST be "TimeZone".
- o tzId: "String" (mandatory).  
The TZID property from iCalendar.
- o updated: "UTCDateTime" (optional)  
The LAST-MODIFIED property from iCalendar.
- o url: "String" (optional)  
The TZURL property from iCalendar.
- o validUntil: "UTCDateTime" (optional)  
The TZUNTIL property from iCalendar specified in [RFC7808].
- o aliases: "String[Boolean]" (optional)

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

- o standard: "TimeZoneRule[]" (optional)

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

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

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

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

- o @type: "String" (mandatory)

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

- o start: "LocalDateTime" (mandatory)

The DTSTART property from iCalendar.

- o offsetFrom: "String" (mandatory)

The TZOFFSETFROM property from iCalendar.

- o offsetTo: "String" (mandatory)

The TZOFFSETTO property from iCalendar.

- o recurrenceRules: "RecurrenceRule[]" (optional)

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

- o recurrenceOverrides: "LocalDateTime[PatchObject]" (optional)

Maps the RDATE properties from iCalendar. The set is represented as an object, with the keys being the recurrence dates. The patch object MUST be the empty JSON object ({}).

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

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

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

## 5. Type-specific JSCalendar Properties

### 5.1. JSEvent Properties

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

#### 5.1.1. start

Type: "LocalDateTime" (mandatory).

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

#### 5.1.2. duration

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

The zero or positive duration of the event in the event's start time zone. The end time of an event can be found by adding the duration to the event's start time.

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

#### 5.1.3. status

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

The scheduling status (Section 4.4) of a JSEvent. If set, it MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- o "confirmed": Indicates the event is definitely happening.
- o "cancelled": Indicates the event has been cancelled.

- o "tentative": Indicates the event may happen.

## 5.2. JSTask Properties

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

### 5.2.1. due

Type: "LocalDateTime" (optional).

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

### 5.2.2. start

Type: "LocalDateTime" (optional).

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

### 5.2.3. estimatedDuration

Type: "Duration" (optional).

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

### 5.2.4. percentComplete

Type: "UnsignedInt" (optional).

Represents the percent completion of the task overall. The property value MUST be a positive integer between 0 and 100.

### 5.2.5. progress

Type: "String" (optional).

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

- o "completed": if the "progress" property value of all participants is "completed".
- o "failed": if at least one "progress" property value of a participant is "failed".

- o "in-process": if at least one "progress" property value of a participant is "in-process".
- o "needs-action": If none of the other criteria match.

If set, it MUST be one of the following values, another value registered in the IANA JSCalendar Enum Values registry, or a vendor-specific value (see Section 3.3):

- o "needs-action": Indicates the task needs action.
- o "in-process": Indicates the task is in process.
- o "completed": Indicates the task is completed.
- o "failed": Indicates the task failed.
- o "cancelled": Indicates the task was cancelled.

#### 5.2.6. progressUpdated

Type: "UTCDateTime" (optional).

Specifies the date/time the "progress" property of either the task overall (Section 5.2.5) or a specific participant (Section 4.4.5) was last updated.

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

#### 5.3. JSGroup Properties

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

- o @type
- o uid
- o prodId



- o created
- o updated
- o title
- o description
- o descriptionContentType
- o links
- o locale
- o keywords
- o categories
- o color
- o timeZones

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

#### 5.3.1. entries

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

A collection of group members. Implementations MUST ignore entries of unknown type.

#### 5.3.2. source

Type: "String" (optional).

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

### 6. Examples

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

### 6.1. Simple event

This example illustrates a simple one-time event. It specifies a one-time event that begins on January 15, 2020 at 1pm New York local time and ends after 1 hour.

```
{
  "@type": "jsevent",
  "uid": "a8df6573-0474-496d-8496-033ad45d7fea",
  "updated": "2020-01-02T18:23:04Z",
  "title": "Some event",
  "start": "2020-01-15T13:00:00",
  "timeZone": "America/New_York",
  "duration": "PT1H"
}
```

### 6.2. Simple task

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

```
{
  "@type": "jstask",
  "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
  "updated": "2020-01-09T14:32:01Z",
  "title": "Do something"
}
```

### 6.3. Simple group

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

```

{
  "@type": "jsgroup",
  "uid": "bf0ac22b-4989-4caf-9ebd-54301b4ee51a",
  "updated": "2020-01-15T18:00:00Z",
  "name": "A simple group",
  "entries": [{
    "@type": "jsevent",
    "uid": "a8df6573-0474-496d-8496-033ad45d7fea",
    "updated": "2020-01-02T18:23:04Z",
    "title": "Some event",
    "start": "2020-01-15T13:00:00",
    "timeZone": "America/New_York",
    "duration": "PT1H"
  },
  {
    "@type": "jstask",
    "uid": "2a358cee-6489-4f14-a57f-c104db4dc2f2",
    "updated": "2020-01-09T14:32:01Z",
    "title": "Do something"
  }]
}

```

#### 6.4. All-day event

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

```

{
  "...": "",
  "title": "April Fool's Day",
  "showWithoutTime": true,
  "start": "1900-04-01T00:00:00",
  "duration": "P1D",
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "yearly"
  }]
}

```

#### 6.5. Task with a due date

This example illustrates a task with a due date. It is a reminder to buy groceries before 6pm Vienna local time on January 19, 2020. The calendar user expects to need 1 hour for shopping.

```
{
  "...": "",
  "title": "Buy groceries",
  "due": "2020-01-19T18:00:00",
  "timeZone": "Europe/Vienna",
  "estimatedDuration": "PT1H"
}
```

#### 6.6. Event with end time zone

This example illustrates the use of end time zones by use of an international flight. The flight starts on April 1, 2020 at 9am in Berlin local time. The duration of the flight is scheduled at 10 hours 30 minutes. The time at the flight's destination is in the same time zone as Tokyo. Calendar clients could use the end time zone to display the arrival time in Tokyo local time and highlight the time zone difference of the flight. The location names can serve as input for navigation systems.

```
{
  "...": "",
  "title": "Flight XY51 to Tokyo",
  "start": "2020-04-01T09:00:00",
  "timeZone": "Europe/Berlin",
  "duration": "PT10H30M",
  "locations": {
    "418d0b9b-b656-4b3c-909f-5b149ca779c9": {
      "@type": "Location",
      "rel": "start",
      "name": "Frankfurt Airport (FRA)"
    },
    "c2c7ac67-dc13-411e-a7d4-0780fb61fb08": {
      "@type": "Location",
      "rel": "end",
      "name": "Narita International Airport (NRT)",
      "timeZone": "Asia/Tokyo"
    }
  }
}
```

#### 6.7. Floating-time event (with recurrence)

This example illustrates the use of floating time. Since January 1, 2020, a calendar user blocks 30 minutes every day to practice Yoga at 7am local time, in whatever time zone the user is located on that date.

```
{
  "...": "",
  "title": "Yoga",
  "start": "2020-01-01T07:00:00",
  "duration": "PT30M",
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "daily"
  }]
}
```

#### 6.8. Event with multiple locations and localization

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

```
{
  "...": "",
  "title": "Live from Music Bowl: The Band",
  "description": "Go see the biggest music event ever!",
  "locale": "en",
  "start": "2020-07-04T17:00:00",
  "timeZone": "America/New_York",
  "duration": "PT3H",
  "locations": {
    "c0503d30-8c50-4372-87b5-7657e8e0fedd": {
      "@type": "Location",
      "name": "The Music Bowl",
      "description": "Music Bowl, Central Park, New York",
      "coordinates": "geo:40.7829,-73.9654"
    }
  },
  "virtualLocations": {
    "1": {
      "@type": "VirtualLocation",
      "name": "Free live Stream from Music Bowl",
      "uri": "https://stream.example.com/the_band_2020"
    }
  },
  "localizations": {
    "de": {
      "title": "Live von der Music Bowl: The Band!",
      "description": "Schau dir das groesste Musikereignis an!",
      "virtualLocations/1/name": "Gratis Live-Stream aus der Music Bowl"
    }
  }
}
```

### 6.9. Recurring event with overrides

This example illustrates the use of recurrence overrides. A math course at a University is held for the first time on January 8, 2020 at 9am London time and occurs every week until June 24, 2020. Each lecture lasts for one hour and 30 minutes and is located at the Mathematics department. This event has exceptional occurrences: at the last occurrence of the course is an exam, which lasts for 2 hours and starts at 10am. Also, the location of the exam differs from the usual location. On April 1 no course is held. On January 7 at 2pm is an optional introduction course, that occurs before the first regular lecture.

```

{
  "...": "",
  "title": "Calculus I",
  "start": "2020-01-08T09:00:00",
  "timeZone": "Europe/London",
  "duration": "PT1H30M",
  "locations": {
    "0dfb8ace-aad1-4734-b3b4-a2fe3d6ae1c5": {
      "@type": "Location",
      "title": "Math lab room 1",
      "description": "Math Lab I, Department of Mathematics"
    }
  },
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "weekly",
    "until": "2020-06-24T09:00:00"
  }],
  "recurrenceOverrides": {
    "2020-01-07T14:00:00": {
      "title": "Introduction to Calculus I (optional)"
    },
    "2020-04-01T09:00:00": {
      "excluded": true
    },
    "2020-06-25T09:00:00": {
      "title": "Calculus I Exam",
      "start": "2020-06-25T10:00:00",
      "duration": "PT2H",
      "locations": {
        "84d639ca-37ac-4a86-81e5-9bbba8eb4053": {
          "@type": "Location",
          "title": "Big Auditorium",
          "description": "Big Auditorium, Other Road"
        }
      }
    }
  }
}

```

#### 6.10. Recurring event with participants

This example illustrates scheduled events. A team meeting occurs every week since January 8, 2020 at 9am Johannesburg time. The event owner also chairs the event. Participants meet in a virtual meeting room. An attendee has accepted the invitation, but on March 4, 2020 he is unavailable and declined participation for this occurrence.

```
{
  "...": "",
  "title": "FooBar team meeting",
  "start": "2020-01-08T09:00:00",
  "timeZone": "Africa/Johannesburg",
  "duration": "PT1H",
  "virtualLocations": {
    "3f41b47b-a5eb-494f-90eb-19d279486d84": {
      "@type": "VirtualLocation",
      "name": "ChatMe meeting room",
      "uri": "https://chatme.example.com?id=1234567&pw=a8a24627b63d396e"
    }
  },
  "recurrenceRules": [{
    "@type": "RecurrenceRule",
    "frequency": "weekly"
  }],
  "replyTo": {
    "imip": "mailto:f245f875-7f63-4a5e-a2c8@schedule.example.com"
  },
  "participants": {
    "dG9tQGZvb2Jhci5x1LmNvbQ": {
      "@type": "Participant",
      "name": "Tom Tool",
      "email": "tom@foobar.example.com",
      "sendTo": {
        "imip": "mailto:tom@calendar.example.com"
      },
      "participationStatus": "accepted",
      "roles": {
        "attendee": true
      }
    },
    "em9lQGZvb2GFtcGx1LmNvbQ": {
      "@type": "Participant",
      "name": "Zoe Zelda",
      "email": "zoe@foobar.example.com",
      "sendTo": {
        "imip": "mailto:zoe@foobar.example.com"
      },
      "participationStatus": "accepted",
      "roles": {
        "owner": true,
        "attendee": true,
        "chair": true
      }
    }
  }
},
```



```
"recurrenceOverrides": {  
  "2020-03-04T09:00:00": {  
    "participants/dG9tQGZvb2Jhci5x1LmNvbQ/participationStatus":  
      "declined"  
  }  
}
```

## 7. Security Considerations

Calendaring and scheduling information is very privacy-sensitive. It can reveal the social network of a user; location information of this user and those in their social network; identity and credentials information; and the patterns of behavior of the user in both the physical and cyber realm. Additionally, calendar events and tasks can could influence the physical location of a user or their cyber behavior within a known time window. Its transmission and storage must be done carefully to protect it from possible threats, such as eavesdropping, replay, message insertion, deletion, modification, and on-path attacks.

The data being stored and transmitted may be used in systems with real world consequences. For example, a home automation system may turn an alarm on and off. Or a coworking space may charge money to the organiser of an event that books one of their meeting rooms. Such systems must be careful to authenticate all data they receive to prevent them from being subverted, and ensure the change comes from an authorized entity.

This document just defines the data format; such considerations are primarily the concern of the API or method of storage and transmission of such files.

### 7.1. Expanding Recurrences

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

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

Events recur in the event's time zone. If the user is in a different time zone, daylight saving transitions may cause an event that normally occurs at, for example, 9am to suddenly shift an hour

earlier. This may be used in an attempt to cause a participant to miss an important meeting. User agents must be careful to translate date-times correctly between time zones and may wish to call out unexpected changes in the time of a recurring event.

## 7.2. JSON Parsing

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

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

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

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

## 7.3. URI Values

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

Fetching remote resources carries inherent risks. Connections must only be allowed on well known ports, using allowed protocols (generally just HTTP/HTTPS on their default ports). The URL must be resolved externally and not allowed to access internal resources. Connecting to an external source reveals IP (and therefore generally location) information.

A maliciously constructed JSCalendar object may contain a very large number of URIs. In the case of published calendars with a large

number of subscribers, such objects could be widely distributed. Implementations should be careful to limit the automatic fetching of linked resources to reduce the risk of this being an amplification vector for a denial-of-service attack.

#### 7.4. Spam

Calendar systems may receive JSCalendar files from untrusted sources, in particular as attachments to emails. This can be a vector for an attacker to inject spam into a user's calendar. This may confuse, annoy, and mislead users, or overwhelm their calendar with bogus events, preventing them from seeing legitimate ones.

Heuristic, statistical or machine-learning-based filters can be effective in filtering out spam. Authentication mechanisms such as DKIM [RFC6376] can help establish the source of messages and associate the data with existing relationships (such as an address book contact). Misclassifications are always possible, however, and providing a mechanism for users to quickly correct this is advised.

Confusable unicode characters may be used to trick a user into trusting a JSCalendar file that appears to come from a known contact but is actually from a similar-looking source controlled by an attacker.

#### 7.5. Duplication

It is important for calendar systems to maintain the UID of an event when updating it to avoid unexpected duplication of events. Consumers of the data may not remove the previous version of the event if it has a different UID. This can lead to a confusing situation for the user, with many variations of the event and no indication of which one is correct. Care must be taken by consumers of the data to remove old events where possible to avoid an accidental denial-of-service attack due to the volume of data.

#### 7.6. Time Zones

Events recur in a particular time zone. When this differs from the user's current time zone, it may unexpectedly cause an occurrence to shift in time for that user due to a daylight savings change in the event's time zone. A maliciously crafted event could attempt to confuse users with such an event to ensure a meeting is missed.

## 8. IANA Considerations

### 8.1. Media Type Registration

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

Type name: application

Subtype name: jscalendar+json

Required parameters: type

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

Optional parameters: none

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

Security considerations: See Section 7 of this document.

Interoperability considerations: While JSCalendar is designed to avoid ambiguities as much as possible, when converting objects from other calendar formats to/from JSCalendar it is possible that differing representations for the same logical data might arise, or ambiguities in interpretation. The semantic equivalence of two JSCalendar objects may be determined differently by different applications, for example where URL values differ in case between the two objects.

Published specification: This specification.

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

Fragment identifier considerations: A JSON Pointer fragment identifier may be used, as defined in [RFC6901], Section 6.

Additional information:

Magic number(s): N/A

File extensions(s): N/A

Macintosh file type code(s): N/A

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

Intended usage: COMMON

Restrictions on usage: N/A

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

Change controller: IETF

## 8.2. Creation of "JSCalendar Properties" Registry

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

This registry follows the Expert Review process ([RFC8126], Section 4.5). If the "intended use" field is "common", sufficient documentation is required to enable interoperability. Preliminary community review for this registry is optional but strongly encouraged.

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

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

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

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

#### 8.2.1. Preliminary Community Review

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

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

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

#### 8.2.2. Submit Request to IANA

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

#### 8.2.3. Designated Expert Review

The primary concern of the designated expert (DE) is preventing name collisions and encouraging the submitter to document security and privacy considerations. For a common-use registration, the DE is expected to confirm that suitable documentation, as described in Section 4.6 of [RFC8126], is available to ensure interoperability. That documentation will usually be in an RFC, but simple definitions are likely to use a web/wiki page, and if a sentence or two is deemed sufficient it could be described in the registry itself. The DE should also verify that the property name does not conflict with work that is active or already published within the IETF. A published specification is not required for reserved or obsolete registrations.

The DE will either approve or deny the registration request and publish a notice of the decision to the Calext WG mailing list or its successor, as well as inform IANA. A denial notice must be justified by an explanation, and, in the cases where it is possible, concrete suggestions on how the request can be modified so as to become acceptable should be provided.

#### 8.2.4. Change Procedures

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

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

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

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

#### 8.2.5. JSCalendar Properties Registry Template

- o Property Name: The name of the property. The property name MUST NOT already be registered for any of the object types listed in the "Property Context" field of this registration. Other object types MAY already have registered a different property with the same name, however the same name SHOULD only be used when the semantics are analogous.
- o Property Type: The type of this property, using type signatures as specified in Section 1.3. The property type MUST be registered in the Type Registry.
- o Property Context: A comma-separated list of JSCalendar object types this property is allowed on.
- o Reference or Description: A brief description or RFC number and section reference where the property is specified (omitted for "reserved" property names).
- o Intended Use: Common, reserved, or obsolete.
- o Change Controller: Who may request a change to this entry's definition ("IETF" for IETF-stream RFCs).

#### 8.2.6. Initial Contents for the JSCalendar Properties Registry

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

Property Name	Property Type	Property Context	Reference or Description
@type	String	JSEvent, JSTask, JSGroup, AbsoluteTrigger, Alert, Link, Location, NDay, OffsetTrigger, Participant, RecurrenceRule, Relation, TimeZone, TimeZoneRule, VirtualLocation	Section 4.1.1, Section 4.5.2, Section 1.4.11, Section 4.2.5, Section 4.4.5, Section 4.3.2, Section 4.1.3, Section 4.7.2, Section 4.7.2, Section 4.2.6
acknowledged	UTCDateTime	Alert	Section 4.5.2
action	String	Alert	Section 4.5.2
alerts	Id[Alert]	JSEvent, JSTask	Section 4.5.2
aliases	String[Boolean]	TimeZone	Section 4.7.2
byDay	NDay[]	Recurrence Rule	Section 4.3.2
byHour	UnsignedInt[]	Recurrence Rule	Section 4.3.2
byMinute	UnsignedInt[]	Recurrence Rule	Section 4.3.2



byMonth	String[]	Recurrence Rule	Section 4.3.2
byMonthDay	Int []	Recurrence Rule	Section 4.3.2
bySecond	UnsignedInt []	Recurrence Rule	Section 4.3.2
bySetPosition	Int []	Recurrence Rule	Section 4.3.2
byWeekNo	Int []	Recurrence Rule	Section 4.3.2
byYearDay	Int []	Recurrence Rule	Section 4.3.2
categories	String[Boolean]	JSEvent, JSTask, JSGroup	Section 4.2.10
cid	String	Link	Section 1.4.11
color	String	JSEvent, JSTask, JSGroup	Section 4.2.11
comments	String[]	TimeZoneRule	Section 4.7.2
contentType	String	Link	Section 1.4.11
coordinates	String	Location	Section 4.2.5
count	UnsignedInt	Recurrence Rule	Section 4.3.2
created	UTCDateTime	JSEvent, JSTask, JSGroup	Section 4.1.5
day	String	NDay	Section 4.3.2

daylight	TimeZoneRule	TimeZone	Section 4.7.2
delegatedFrom	String[Boolean]	Participant	Section 4.4.5
delegatedTo	String[Boolean]	Participant	Section 4.4.5
description	String	JSEvent, JSTask, Location, Participant, Virtual Location	Section 4.2.2, Section 4.2.5, Section 4.4.5, Section 4.2.6
descriptionContentType	String	JSEvent, JSTask	Section 4.2.3
display	String	Link	Section 1.4.11
due	LocalDateTime	JSTask	Section 5.2.1
duration	Duration	JSEvent	Section 5.1.2
email	String	Participant	Section 4.4.5
entries	(JSTask JSEvent) []	JSGroup	Section 5.3.1
estimatedDuration	Duration	JSTask	Section 5.2.3
excluded	Boolean	JSEvent, JSTask	Section 4.3.5
excludedRecurrenceRules	RecurrenceRule[]	JSEvent, JSTask	Section 4.3.3
expectReply	Boolean	Participant	Section 4.4.5

firstDayOfWeek	String	Recurrence Rule	Section 4.3.2
freeBusyStatus	String	JSEvent, JSTask	Section 4.4.2
frequency	String	Recurrence Rule	Section 4.3.2
href	String	Link	Section 1.4.11
interval	UnsignedInt	Recurrence Rule	Section 4.3.2
invitedBy	String	Participant	Section 4.4.5
keywords	String[Boolean]	JSEvent, JSTask, JSGroup	Section 4.2.9
kind	String	Participant	Section 4.4.5
language	String	Participant	Section 4.4.5
links	Id[Link]	JSGroup, JSEvent, JSTask, Location, Participant	Section 4.2.7, Section 4.2.5, Section 4.4.5
locale	String	JSGroup, JSEvent, JSTask	Section 4.2.8
localizations	String[PatchObject]	JSEvent, JSTask	Section 4.6.1
locationId	String	Participant	Section 4.4.5
locations	Id[Location]	JSEvent, JSTask	Section 4.2.5

locationTypes	String[Boolean]	Location	Section 4.2.5
memberOf	String[Boolean]	Participant	Section 4.4.5
method	String	JSEvent, JSTask	Section 4.1.8
name	String	Location, VirtualLocation, Participant	Section 4.2.5, Section 4.2.6, Section 4.4.5
names	String[Boolean]	TimeZoneRule	Section 4.7.2
nthOfPeriod	Int	NDay	Section 4.3.2
offset	SignedDuration	OffsetTrigger	Section 4.5.2
offsetFrom	UTCDateTime	TimeZoneRule	Section 4.7.2
offsetTo	UTCDateTime	TimeZoneRule	Section 4.7.2
participants	Id[Participant]	JSEvent, JSTask	Section 4.4.5
participationComment	String	Participant	Section 4.4.5
participationStatus	String	Participant	Section 4.4.5
percentComplete	UnsignedInt	JSTask, Participant	Section 5.2.4
priority	Int	JSEvent, JSTask	Section 4.4.1
privacy	String	JSEvent,	Section

			JSTask	4.4.3
prodId	String		JSEvent, JSTask, JSGroup	Section 4.1.4
progress	String		JSTask, Participant	Section 5.2.5
progressUpdated	UTCDateTime		JSTask, Participant	Section 5.2.6
recurrenceId	LocalDateTime		JSEvent, JSTask	Section 4.3.1
recurrenceOverrides	LocalDateTime[PatchObject]		JSEvent, JSTask, TimeZoneRule	Section 4.3.4, Section 4.7.2
recurrenceRules	RecurrenceRule[]		JSEvent, JSTask, TimeZoneRule	Section 4.3.2, Section 4.7.2
rel	String		Link	Section 1.4.11
relatedTo	String[Relation]		JSEvent, JSTask, Alert	Section 4.1.3, Section 4.5.2
relation	String[Boolean]		Relation	Section 1.4.10
relativeTo	String		OffsetTrigger, Location	Section 4.5.2, Section 4.2.5
replyTo	String[String]		JSEvent, JSTask	Section 4.4.4
roles	String[Boolean]		Participant	Section 4.4.5
rscale	String		Recurrence	Section

		Rule	4.3.2
standard	TimeZoneRule	TimeZone	Section 4.7.2
start	LocalDateTime	TimeZoneRule	Section 4.7.2
scheduleAgent	String	Participant	Section 4.4.5
scheduleForceSend	Boolean	Participant	Section 4.4.5
scheduleSequence	UnsignedInt	Participant	Section 4.4.5
scheduleStatus	String[]	Participant	Section 4.4.5
scheduleUpdated	UTCDateTime	Participant	Section 4.4.5
sendTo	String[String]	Participant	Section 4.4.5
sequence	UnsignedInt	JSEvent, JSTask	Section 4.1.7
showWithoutTime	Boolean	JSEvent, JSTask	Section 4.2.4
size	UnsignedInt	Link	Section 1.4.11
skip	String	Recurrence Rule	Section 4.3.2
source	String	JSGroup	Section 5.3.2
start	LocalDateTime	JSEvent, JSTask	Section 5.1.1, Section 5.2.2
status	String	JSEvent	Section 5.1.3

timeZone	TimeZoneId null	JSEvent, JSTask, Location	Section 4.7.1, Section 4.2.5
timeZones	TimeZoneId[TimeZone]	JSEvent, JSTask	Section 4.7.2
title	String	JSEvent, JSTask, JSGroup, Link	Section 4.2.1
trigger	OffsetTrigger AbsoluteTrigger UnknownTrigger	Alert	Section 4.5.2
tzId	String	TimeZone	Section 4.7.2
uid	String	JSEvent, JSTask, JSGroup	Section 4.1.2
until	LocalDateTime	Recurrence Rule	Section 4.3.2
updated	UTCDateTime	JSEvent, JSTask, JSGroup	Section 4.1.6
uri	String	VirtualLocation	Section 4.2.6
url	String	TimeZone	Section 4.7.2
useDefaultAlerts	Boolean	JSEvent, JSTask	Section 4.5.1
validUntil	UTCDateTime	TimeZone	Section 4.7.2
virtualLocations	Id[VirtualLocation]	JSEvent, JSTask	Section 4.2.6
when	UTCDateTime	AbsoluteTrigger	Section 4.5.2

+-----+-----+-----+-----+

Table 1

### 8.3. Creation of "JSCalendar Types" Registry

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

#### 8.3.1. JSCalendar Types Registry Template

- o Type Name: The name of the type.
- o Reference or Description: A brief description or RFC number and section reference where the Type is specified (may be omitted for "reserved" type names).
- o Intended Use: Common, reserved, or obsolete.
- o Change Controller: Who may request a change to this entry's definition ("IETF" for IETF-stream RFCs).

#### 8.3.2. Initial Contents for the JSCalendar Types Registry

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

Type Name	Reference or Description
Alert	Section 4.5.2
Boolean	Section 1.3
Duration	Section 1.4.6
Id	Section 1.4.1
Int	Section 1.4.2
LocalDateTime	Section 1.4.5
Link	Section 1.4.11



Location	Section 4.2.5
NDay	Section 4.3.2
Number	Section 1.3
Participant	Section 4.4.5
PatchObject	Section 1.4.9
RecurrenceRule	Section 4.3.2
Relation	Section 1.4.10
SignedDuration	Section 1.4.7
String	Section 1.3
TimeZone	Section 4.7.2
TimeZoneId	Section 1.4.8
TimeZoneRule	Section 4.7.2
UnsignedInt	Section 1.4.3
UTCDateTime	Section 1.4.4
VirtualLocation	Section 4.2.6

Table 2

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

The IANA will create the "JSCalendar Enum Values" registry to allow interoperable extension of semantics for properties with enumerable values. Each such property will have a subregistry of allowed values. The registration process for a new enum value or adding a new enumerable property is the same as for the JSCalendar Properties registry, as defined in Section 8.2.

##### 8.4.1. JSCalendar Enum Property Template

This template is for adding a subregistry for a new enumerable property to the JSCalendar Enum registry.

- o **Property Name:** the name(s) of the property or properties where these values may be used. This MUST be registered in the JSCalendar Properties registry.
- o **Context:** the list of allowed object types where the property or properties may appear, as registered in the JSCalendar Properties registry. This disambiguates where there may be two distinct properties with the same name in different contexts.
- o **Change Controller:** ("IETF" for properties defined in IETF-stream RFCs).
- o **Initial Contents:** The initial list of defined values for this enum, using the template defined in Section 8.4.2. A subregistry will be created with these values for this property name/context tuple.

#### 8.4.2. JSCalendar Enum Value Template

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

- o **Enum Value:** The verbatim value of the enum.
- o **Reference or Description:** A brief description or RFC number and section reference for the semantics of this value.

#### 8.4.3. Initial Contents for the JSCalendar Enum Values registry

For each subregistry created in this section, all RFC section references are for this document.

-----

Property Name: action

Context: Alert

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
display	Section 4.5.2
email	Section 4.5.2

Table 3

---

Property Name: display

Context: Link

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
badge	Section 1.4.11
graphic	Section 1.4.11
fullsize	Section 1.4.11
thumbnail	Section 1.4.11

Table 4

---

Property Name: freeBusyStatus

Context: JSEvent, JSTask

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
free	Section 4.4.2
busy	Section 4.4.2

Table 5

---

Property Name: kind

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
individual	Section 4.4.5
group	Section 4.4.5
resource	Section 4.4.5
location	Section 4.4.5

Table 6

---

Property Name: participationStatus

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
needs-action	Section 4.4.5
accepted	Section 4.4.5
declined	Section 4.4.5
tentative	Section 4.4.5
delegated	Section 4.4.5

Table 7

---

Property Name: `privacy`

Context: `JSEvent`, `JSTask`

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
public	Section 4.4.3
private	Section 4.4.3
secret	Section 4.4.3

Table 8

---

Property Name: `progress`

Context: `JSTask`, `Participant`

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
needs-action	Section 5.2.5
in-process	Section 5.2.5
completed	Section 5.2.5
failed	Section 5.2.5
cancelled	Section 5.2.5

Table 9

---

Property Name: relation

Context: Relation

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
first	Section 1.4.10
next	Section 1.4.10
child	Section 1.4.10
parent	Section 1.4.10

Table 10

---

Property Name: relativeTo

Context: OffsetTrigger, Location

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
start	Section 4.5.2
end	Section 4.5.2

Table 11

---

Property Name: roles

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
owner	Section 4.4.5
attendee	Section 4.4.5
optional	Section 4.4.5
informational	Section 4.4.5
chair	Section 4.4.5
contact	Section 4.4.5

Table 12

---

Property Name: scheduleAgent

Context: Participant

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
server	Section 4.4.5
client	Section 4.4.5
none	Section 4.4.5

Table 13

---

Property Name: status

Context: JSEvent

Change Controller: IETF

Initial Contents:

Enum Value	Reference or Description
confirmed	Section 5.1.3
cancelled	Section 5.1.3
tentative	Section 5.1.3

Table 14

## 9. Acknowledgments

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

## 10. References



## 10.1. Normative References

- [CLDR] "Unicode Common Locale Data Repository",  
<<http://cldr.unicode.org/>>.
- [COLORS] "CSS Color Module", <<https://www.w3.org/TR/css-color-3/>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, DOI 10.17487/RFC2392, August 1998, <<https://www.rfc-editor.org/info/rfc2392>>.
- [RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, DOI 10.17487/RFC2397, August 1998, <<https://www.rfc-editor.org/info/rfc2397>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC4589] Schulzrinne, H. and H. Tschofenig, "Location Types Registry", RFC 4589, DOI 10.17487/RFC4589, July 2006, <<https://www.rfc-editor.org/info/rfc4589>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.

- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <<https://www.rfc-editor.org/info/rfc5545>>.
- [RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <<https://www.rfc-editor.org/info/rfc5546>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <<https://www.rfc-editor.org/info/rfc5870>>.
- [RFC6047] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 6047, DOI 10.17487/RFC6047, December 2010, <<https://www.rfc-editor.org/info/rfc6047>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <<https://www.rfc-editor.org/info/rfc6901>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC7529] Daboo, C. and G. Yakushev, "Non-Gregorian Recurrence Rules in the Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 7529, DOI 10.17487/RFC7529, May 2015, <<https://www.rfc-editor.org/info/rfc7529>>.
- [RFC7808] Douglass, M. and C. Daboo, "Time Zone Data Distribution Service", RFC 7808, DOI 10.17487/RFC7808, March 2016, <<https://www.rfc-editor.org/info/rfc7808>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <<https://www.rfc-editor.org/info/rfc8288>>.
- [TZDB] "IANA Time Zone Database", <<https://www.iana.org/time-zones>>.

## 10.2. Informative References

- [LINKRELS] "IANA Link Relation Types", <<https://www.iana.org/assignments/link-relations/link-relations.xhtml>>.
- [LOCATIONTYPES] "IANA Location Types Registry", <<https://www.iana.org/assignments/location-type-registry/location-type-registry.xhtml>>.
- [MEDIATYPES] "IANA Media Types", <<https://www.iana.org/assignments/media-types/media-types.xhtml>>.
- [RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <<https://www.rfc-editor.org/info/rfc6376>>.
- [RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", RFC 7265, DOI 10.17487/RFC7265, May 2014, <<https://www.rfc-editor.org/info/rfc7265>>.

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

Authors' Addresses

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

Email: [neilj@fastmailteam.com](mailto: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](mailto:rsto@fastmailteam.com)  
URI: <https://www.fastmail.com>