

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 19, 2013

P. Kewisch
Mozilla
C. Daboo
Apple, Inc.
M. Douglass
RPI
January 15, 2013

jCal: The JSON format for iCalendar
draft-kewisch-et-al-icalendar-in-json-01

Abstract

This specification defines "jCal", a JSON format for iCalendar 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 <http://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 July 19, 2013.

Copyright Notice

Copyright (c) 2013 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 (<http://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
2.	Conventions Used in This Document	4
3.	Converting from iCalendar to jCal	5
3.1.	Pre-processing	5
3.2.	iCalendar stream (RFC5545 section 3.4)	5
3.3.	Components (RFC5545 section 3.6)	7
3.4.	Properties (RFC5545 section 3.7 and 3.8)	7
3.4.1.	Special Cases for Properties	10
3.4.1.1.	Multi-valued Properties (RFC5545 Section 3.1.2)	10
3.4.1.2.	GEO Property (RFC5545 Section 3.8.1.6)	10
3.4.1.3.	REQUEST-STATUS Property (RFC5545 Section 3.8.8.3)	11
3.5.	Parameters (RFC5545 section 3.2)	11
3.5.1.	VALUE parameter	13
3.5.2.	Multi-value Parameters	13
3.6.	Values (RFC5545 section 3.3)	14
3.6.1.	Binary (RFC5545 section 3.3.1)	14
3.6.2.	Boolean (RFC5545 section 3.3.2)	15
3.6.3.	Calendar User Address (RFC5545 section 3.3.3)	15
3.6.4.	Date (RFC5545 section 3.3.4)	15
3.6.5.	Date-Time (RFC5545 section 3.3.5)	15
3.6.6.	Duration (RFC5545 section 3.3.6)	16
3.6.7.	Float (RFC5545 section 3.3.7)	16
3.6.8.	Integer (RFC5545 section 3.3.8)	16
3.6.9.	Period of Time (RFC5545 section 3.3.9)	17
3.6.10.	Recurrence Rule (RFC5545 section 3.3.10)	17
3.6.11.	Text (RFC5545 section 3.3.11)	17
3.6.12.	Time (RFC5545 section 3.3.12)	17
3.6.13.	URI (RFC5545 section 3.3.13)	18
3.6.14.	UTC Offset (RFC5545 section 3.3.14)	18
3.7.	Extensions	19
4.	Converting from jCal into iCalendar	19
5.	Handling Unrecognized Properties or Parameters	19
6.	Security Considerations	21
7.	IANA Considerations	21
8.	Acknowledgments	22
9.	References	22
9.1.	Normative References	22
9.2.	Informative References	23
Appendix A.	ABNF Schema	23
Appendix B.	Examples	25
B.1.	Example 1	25
B.1.1.	iCalendar Data	25
B.1.2.	jCal Data	25
B.2.	Example 2	26
B.2.1.	iCalendar Data	26

[B.2.2.](#) jCal Data 27

[Appendix C.](#) Change History (to be removed prior to
publication as an RFC) 29

1. Introduction

The iCalendar data format [[RFC5545](#)] is a widely deployed interchange format for calendaring and scheduling data. While many applications and services consume and generate calendar data, iCalendar is a specialized format that requires its own parser/generator. In contrast, JSON-based formats as defined in [[RFC4627](#)] are the native format for JavaScript widgets and libraries and it is appropriate to have a standard form of calendar data that is easier to work with than iCalendar.

The purpose of this specification is to define "jCal", a JSON format for iCalendar data. jCal is defined as a straightforward mapping into JSON from iCalendar, so that iCalendar data can be converted to JSON, and then back to iCalendar, without losing any semantic meaning in the data. Anyone creating jCal calendar data according to this specification will know that their data can be converted to a valid iCalendar representation as well.

The key design considerations are essentially the same as those for [[RFC6321](#)], that is:

Round-tripping (converting an iCalendar instance to jCal and back) will give the same semantic result as the starting point. For example, all components, properties and property parameters are guaranteed to be preserved, with the exception of those which have default values.

Ordering of elements will not necessarily be preserved.

Preserve the semantics of the iCalendar data. While a simple consumer can easily browse the calendar data in jCal, a full understanding of iCalendar is still required in order to modify and/or fully comprehend the calendar data.

Ability to handle many extensions to the underlying iCalendar specification without requiring an update to this document.

2. Conventions Used in This Document

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

The underlying format used for jCal is JSON. Consequently, the terms "object" and "array" as well as the four primitive types are to be interpreted as described in [Section 1 of \[RFC4627\]](#).

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.

3. Converting from iCalendar to jCal

This section describes how iCalendar data is converted to jCal using a simple mapping between the iCalendar data model and JSON elements.

3.1. Pre-processing

iCalendar uses a line folding mechanism to limit lines of data to a maximum line length (typically 72 characters) to ensure maximum likelihood of preserving data integrity as it is transported via various means (e.g., email) - see [Section 3.1 of \[RFC5545\]](#). Prior to converting iCalendar data into jCal all folded lines MUST be unfolded.

iCalendar data uses an "escape" character sequence for text values and property parameter values. When such text elements are converted into jCal the escaping MUST be removed.

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

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

When base64 encoding and decoding is used, it MUST conform to [Section 4 of \[RFC4648\]](#), which is the base64 method used in [\[RFC5545\]](#).

One key difference in the formatting of values used in iCalendar and jCal is that in jCal the specification uses date/time and utc-offset values aligned with the complete representation, extended format of [\[ISO.8601.2004\]](#).

3.2. iCalendar stream ([RFC5545 section 3.4](#))

At the top level of the iCalendar object model is an "iCalendar stream". This stream encompasses multiple "iCalendar objects". In jCal, the entire stream is represented by an array, where the first element is the string "icalendar" and subsequent elements are

iCalendar objects represented as described in this section.

In the typical case where there is only one iCalendar object, encapsulation inside an "icalendar" array MAY be omitted.

An iCalendar stream can contain one or more iCalendar objects. Each iCalendar object, delimited by "BEGIN:VCALENDAR" and "END:VCALENDAR", is represented by a fixed length array with three elements:

1. The string "vcalendar"
2. An array of jCal properties
3. An array of jCal components

The representation of an iCalendar object in JSON will be named "vcalendar component" throughout this document.

Example:

```
[ "icalendar",
  [ "vcalendar",
    [ /* properties */ ],
    [ /* components */ ]
  ],
  [ "vcalendar",
    [ /* properties */ ],
    [ /* components */ ]
  ],
  ...
]
```

iCalendar objects are comprised of a set of "components", "properties", "parameters" and "values". A "component" can contain other "components" or "properties". A "property" has a value and a set of zero or more "parameters".

An iCalendar object may contain a mix of iCalendar component types, for example vevent objects delimited by "BEGIN:VEVENT" and "END:VEVENT", task objects delimited by "BEGIN:VTOD" and "END:VTOD".

Item	jCal name	jCal Definition
iCalendar Stream	"icalendar"	Appendix A "jcalstream"
VCALENDAR	"vcalendar"	Appendix A "jcalobject"

3.3. Components (RFC5545 section 3.6)

Each calendar component in the "VCALENDAR" object, delimited by "BEGIN" and "END", will be converted to a fixed length array with three fields that have a specific structure:

1. A string with the name of the iCalendar component, but in lowercase.
2. An array of jCal properties as described in [Section 3.4](#).
3. An array of jCal components, representing the sub-components of the component in question.

While the grouping of properties and sub-components does not retain the original order specified in the iCalendar stream, the semantics of a component are preserved.

As an example, the table below shows iCalendar to jCal mappings for current iCalendar components. Any new iCalendar components added in the future will be converted in the same way.

Component	jCal name	RFC5545 Section
VEVENT	vevent	[RFC5545] # 3.6.1
VTODO	vtodo	[RFC5545] # 3.6.2
VJOURNAL	vjournal	[RFC5545] # 3.6.3
VFREEBUSY	vfreebusy	[RFC5545] # 3.6.4
VTIMEZONE	vtimezone	[RFC5545] # 3.6.5
STANDARD	standard	[RFC5545] # 3.6.5
DAYLIGHT	daylight	[RFC5545] # 3.6.5
VALARM	valarm	[RFC5545] # 3.6.6

Example:

```
[ "vevent",
  [ /* Array of jCal properties */ ],
  [ /* Array of jCal sub-components */ ]
]
```

3.4. Properties (RFC5545 section 3.7 and 3.8)

iCalendar properties, whether they apply to the "VCALENDAR" object or to a component, are handled in a consistent way in the jCal format.

Each individual iCalendar property is represented in jCal by an array

with three fixed elements, followed by at one or more additional elements, depending on if the property is a multi-value property as described in [Section 3.1.2 of \[RFC5545\]](#).

The array consists of the following fixed elements:

1. The name of the property as a string, but in lowercase.
2. An object containing the parameters as described in [Section 3.5](#).
3. The type identifier string of the value, in lowercase.

The remaining elements of the array are used for the value of the property. For single-value properties, the array **MUST** have exactly four elements, for multi-valued properties as described in [Section 3.4.1.1](#) there can be any number of additional elements.

The array describing the property can then be inserted into the array designated for properties in any component, including the "vcalendar" component described in [Section 3.3](#).

Example:

```
[ "vevent",
  [
    ["summary", {}, "text", "Meeting with Fred"],
    ["categories", {}, "text", "Meetings", "Work"]
    ...
  ],
  [ /* sub-components */ ]
]
```

The property parameters in the second element of the property array associate a set of parameter names with their respective value. Parameters are further described in [Section 3.5](#).

To allow for a cleaner implementation, the parameter object **MUST** be present even if there are no parameters. In this case, an empty object **MUST** be used.

As an example, the table below shows iCalendar to jCal mappings for current iCalendar properties. Any new iCalendar properties added in the future will be converted in the same way.

Property	jCal name	RFC5545 Section
CALSCALE	calscale	[RFC5545] # 3.7.1

METHOD	method	[RFC5545] # 3.7.2
PROID	prodid	[RFC5545] # 3.7.3
VERSION	version	[RFC5545] # 3.7.4
ATTACH	attach	[RFC5545] # 3.8.1.1
CATEGORIES	categories	[RFC5545] # 3.8.1.2
CLASS	class	[RFC5545] # 3.8.1.3
COMMENT	comment	[RFC5545] # 3.8.1.4
DESCRIPTION	description	[RFC5545] # 3.8.1.5
GEO	geo	[RFC5545] # 3.8.1.6
LOCATION	location	[RFC5545] # 3.8.1.7
PERCENT-COMPLETE	percent-complete	[RFC5545] # 3.8.1.8
PRIORITY	priority	[RFC5545] # 3.8.1.9
RESOURCES	resources	[RFC5545] # 3.8.1.10
STATUS	status	[RFC5545] # 3.8.1.11
SUMMARY	summary	[RFC5545] # 3.8.1.12
COMPLETED	completed	[RFC5545] # 3.8.2.1
DTEND	dtend	[RFC5545] # 3.8.2.2
DUE	due	[RFC5545] # 3.8.2.3
DTSTART	dtstart	[RFC5545] # 3.8.2.4
DURATION	duration	[RFC5545] # 3.8.2.5
FREEBUSY	freebusy	[RFC5545] # 3.8.2.6
TRANSP	transp	[RFC5545] # 3.8.2.7
TZID	tzid	[RFC5545] # 3.8.3.1
TZNAME	tzname	[RFC5545] # 3.8.3.2
TZOFFSETFROM	tzoffsetfrom	[RFC5545] # 3.8.3.3
TZOFFSETTO	tzoffsetto	[RFC5545] # 3.8.3.4
TZURL	tzurl	[RFC5545] # 3.8.3.5
ATTENDEE	attendee	[RFC5545] # 3.8.4.1
CONTACT	contact	[RFC5545] # 3.8.4.2
ORGANIZER	organizer	[RFC5545] # 3.8.4.3
RECURRENCE-ID	recurrence-id	[RFC5545] # 3.8.4.4
RELATED-TO	related-to	[RFC5545] # 3.8.4.5
URL	url	[RFC5545] # 3.8.4.6
UID	uid	[RFC5545] # 3.8.4.7
EXDATE	exdate	[RFC5545] # 3.8.5.1
RDATE	rdate	[RFC5545] # 3.8.5.2
RRULE	rrule	[RFC5545] # 3.8.5.3
ACTION	action	[RFC5545] # 3.8.6.1
REPEAT	repeat	[RFC5545] # 3.8.6.2
TRIGGER	trigger	[RFC5545] # 3.8.6.3
CREATED	created	[RFC5545] # 3.8.7.1
DTSTAMP	dtstamp	[RFC5545] # 3.8.7.2
LAST-MODIFIED	last-modified	[RFC5545] # 3.8.7.3
SEQUENCE	sequence	[RFC5545] # 3.8.7.4
REQUEST-STATUS	request-status	[RFC5545] # 3.8.8.3

3.4.1. Special Cases for Properties

This section describes some properties that have special handling when converting to jCal.

3.4.1.1. Multi-valued Properties (RFC5545 Section 3.1.2)

The following iCalendar properties can have values that consist of a list of "standard" iCalendar values separated by a specific delimiter. In jCal these properties are added as further members of the array describing the property.

Property	jCal name	RFC5545 Section
CATEGORIES	categories	[RFC5545] # 3.8.1.2
RESOURCES	resources	[RFC5545] # 3.8.1.10
FREEBUSY	freebusy	[RFC5545] # 3.8.2.6
EXDATE	exdate	[RFC5545] # 3.8.5.1
RDATE	rdate	[RFC5545] # 3.8.5.2

3.4.1.2. GEO Property (RFC5545 Section 3.8.1.6)

In iCalendar, the "GEO" property value is defined as a semi-colon separated list of two "FLOAT" values, the first representing latitude and the second longitude.

In jCal, the value for the "geo" property is represented like a multi-value property. The first value of the property represents the latitude, the second value represents the longitude.

When converting from jCal to iCalendar, the two values MUST be converted using a semi-colon as the separator character.

Example

```
[ "vevent",
  [
    ["geo", {}, "float", 37.386013, -122.082932 ]
    ...
  ],
  ...
]
```

3.4.1.3. REQUEST-STATUS Property (RFC5545 Section 3.8.8.3)

In iCalendar, the "REQUEST-STATUS" property value is defined as a semi-colon separated list of two or three "TEXT" values. The first represents a code, the second a description, and the third any additional data.

In jCal, the value for the "request-status" property is represented like a multi-value property. The first value of the property corresponds to the code, the second value corresponds to the description and the third value corresponds to the additional data. Each value is represented using a string value. If there is no additional data in the iCalendar value, the last element of the array SHOULD NOT be present.

When converting from jCal to iCalendar, the two or three values MUST be converted using a semi-colon as the separator character.

Example:

```
[ "vevent":  
  [  
    ["request-status", {}, "text", "2.0", "Success" ],  
    ["request-status", {}, "text",  
      "3.7",  
      "Invalid Calendar User",  
      "ATTENDEE:mailto:jsmith@example.org"  
    ],  
    ...  
  ],  
  ...  
]
```

3.5. Parameters (RFC5545 section 3.2)

Property parameters are represented as a JSON object where each key-value pair represents the iCalendar parameter name and its value. The name of the parameter MUST be in lowercase, the original case of the parameter value MUST be preserved.

Each individual iCalendar property parameter is represented in jCal by a key-value pair in the parameters object. The key uses the same name as the iCalendar property parameter, but in lowercase. For example, the "PARTSTAT" property parameter is represented in jCal by the "partstat" key. The case of the parameter value MUST be preserved.

Example:

```
[ "vevent":
  [
    [ "attendee",
      {
        "partstat": "ACCEPTED",
        "rsvp": "TRUE",
        "role": "REQ-PARTICIPANT"
      },
      "cal-address",
      "mailto:jsmith@example.org"
    ],
    [ "summary", {}, "text", "Meeting"],
    ...
  ],
  ...
]
```

As an example, the table below shows iCalendar to jCal mappings for current iCalendar parameters. Any new iCalendar parameters added in the future will be converted in the same way.

Parameter	jCal name	RFC5545 Section
ALTREP	altrep	[RFC5545] # 3.2.1
CN	cn	[RFC5545] # 3.2.2
CUTYPE	cutype	[RFC5545] # 3.2.3
DELEGATED-FROM	delegated-from	[RFC5545] # 3.2.4
DELEGATED-TO	delegated-to	[RFC5545] # 3.2.5
DIR	dir	[RFC5545] # 3.2.6
ENCODING	encoding	[RFC5545] # 3.2.7
FMTTYPE	fmttype	[RFC5545] # 3.2.8
FBTYPE	fbtype	[RFC5545] # 3.2.9
LANGUAGE	language	[RFC5545] # 3.2.10
MEMBER	member	[RFC5545] # 3.2.11
PARTSTAT	partstat	[RFC5545] # 3.2.12
RANGE	range	[RFC5545] # 3.2.13
RELATED	related	[RFC5545] # 3.2.14
RELTYPE	reltype	[RFC5545] # 3.2.15
ROLE	role	[RFC5545] # 3.2.16
RSVP	rsvp	[RFC5545] # 3.2.17
SENT-BY	sent-by	[RFC5545] # 3.2.18
TZID	tzid	[RFC5545] # 3.2.19

3.5.1. VALUE parameter

iCalendar defines a "VALUE" property parameter ([Section 3.2.20 of \[RFC5545\]](#)). This property parameter MUST NOT be added to the parameters object. Instead, the value type is always explicitly mentioned in the third element of the array describing the property. Thus, when converting from iCalendar to jCal, any "VALUE" property parameters are skipped. When converting from jCalendar into iCalendar, the appropriate "VALUE" property parameter MUST be included in the iCalendar property if the value type is not the default value type for that property.

3.5.2. Multi-value Parameters

In [\[RFC5545\]](#), some parameters allow using a COMMA-separated list of values. To ease processing in jCal, the value to such parameters MUST be represented in an array containing the separated values. The array elements MUST be string values. Single-value parameters SHOULD be represented using a single string value, but an array with one element MAY also be used.

DQUOTE characters used to encapsulate the separated values MUST NOT be added to the jCal parameter value.

Example 1:

```
...
["attendee",
 {
   "delegated-to": ["mailto:jdoe@example.org",
                   "mailto:jqpublic@example.org"]
 },
 "cal-address",
 "mailto:jsmith@example.org"
],
...
```

Example 2:

```
...
["attendee",
 {
  "delegated-to": "mailto:jdoh@example.org"
 },
 "cal-address",
 "mailto:jsmith@example.org"
 ],
...
```

As an example, the table below shows a list of iCalendar parameters where COMMA-separated values are used. Any new iCalendar parameters added in the future will be handled in the same way.

Parameter	RFC5545 Section
DELEGATED-FROM	[RFC5545] # 3.2.4
DELEGATED-TO	[RFC5545] # 3.2.5
MEMBER	[RFC5545] # 3.2.11

3.6. Values (RFC5545 section 3.3)

The type of an iCalendar value is explicitly mentioned in the third element of the array describing a jCal property. The actual values of the property can be found in the fourth and following elements of the array.

3.6.1. Binary (RFC5545 section 3.3.1)

Description: iCalendar "BINARY" property values are represented by a property with type identifier "binary". The value is base64 encoded data, conforming to Section 4 of [RFC4648], which is the base64 method used in [RFC5545].

Example:

```
...
["attach", {}, "binary", "SGVsbG8gV29ybGQh"],
...
```

3.6.2. Boolean ([RFC5545 section 3.3.2](#))

Description: iCalendar "BOOLEAN" property values are represented by a property with the type identifier "binary". The value is a boolean JSON value.

Example:

```
...
["x-non-smoking", {}, "boolean", true],
...
```

3.6.3. Calendar User Address ([RFC5545 section 3.3.3](#))

Description: iCalendar "CAL-ADDRESS" property values are represented by a property with the type identifier "cal-address". The value is a string with the URI as described in [\[RFC3986\]](#).

Example:

```
...
["attendee", {}, "cal-address", "mailto:kewisch@example.com"],
...
```

3.6.4. Date ([RFC5545 section 3.3.4](#))

Description: iCalendar "DATE" property values are represented by a property with the type identifier "date". The value is the same date value specified by [\[RFC5545\]](#), but formatted using the [\[ISO.8601.2004\]](#) complete representation, extended format. The textual format specifies a four-digit year, two-digit month, and two-digit day of the month, separated by the "-" character.

Example:

```
...
["dtstart", {}, "date", "2011-05-17"],
...
```

3.6.5. Date-Time ([RFC5545 section 3.3.5](#))

Description: iCalendar "DATE-TIME" property values are represented by a property with the type identifier "date-time". The value is the same date value specified by [\[RFC5545\]](#), but formatted using the [\[ISO.8601.2004\]](#) complete representation, extended format. The same restrictions with respect to leap seconds and timezone offsets as specified in [\[RFC5545\] Section 3.3.5](#) apply.

Example:

```
...
["dtstart", {}, "date-time", "2012-10-17T12:00:00"],
["dtstamp", {}, "date-time", "2012-10-17T12:00:00Z"],
["dtend",
 { "tzid": "Europe/Berlin" },
 "date-time",
 "2011-10-17T13:00:00"
],
...
```

3.6.6. Duration (RFC5545 section 3.3.6)

Description: iCalendar "DURATION" property values are represented by a property with the type identifier "duration". The value is the same duration value specified by [RFC5545] as a string.

Example:

```
...
["duration", {}, "duration", "P1D"],
...
```

3.6.7. Float (RFC5545 section 3.3.7)

Description: iCalendar "FLOAT" property values are represented by a property with the type identifier "float". The value is the floating point number as a JSON primitive number value.

Example:

```
...
["x-grade", {}, "float", 1.3],
...
```

3.6.8. Integer (RFC5545 section 3.3.8)

Description: iCalendar "INTEGER" property values are represented by a property with the type identifier "integer". The value is the floating point number as a JSON primitive number value.

Examples:

```
...
["percent-complete", {}, "integer", 42],
...
```


3.6.9. Period of Time (RFC5545 section 3.3.9)

Description: iCalendar "PERIOD" property values are represented by a property with the type identifier "period". The value is the same period value specified by [RFC5545] as a string. Any date or date-time values contained in the period value MUST be formatted in accordance to the rules for date or date-time values.

Example:

```
...
["freebusy",
 { "fbtype": "FREE" },
 "period",
 "1997-03-08T16:00:00Z/P1D"
],
...
```

3.6.10. Recurrence Rule (RFC5545 section 3.3.10)

Description: iCalendar "RECUR" property values are represented by a property with the type identifier "recur". The value is the same value specified by [RFC5545] as a string.

Example:

```
...
["rrule", {}, "recur", "FREQ=YEARLY;COUNT=5;BYDAY-1SU;BYMONTH=10"],
...
```

3.6.11. Text (RFC5545 section 3.3.11)

Description: iCalendar "TEXT" property values are represented by a property with the type identifier "text". The value is the same text value specified by [RFC5545] as a string.

Example:

```
...
["comment", {}, "text", "hello, world"],
...
```

3.6.12. Time (RFC5545 section 3.3.12)

Description: iCalendar "TIME" property values are represented by a property with the type identifier "time". The value is the same date value specified by [RFC5545], but formatted using the [ISO.8601.2004] complete representation, extended format. The same restrictions with respect to leap seconds, time fractions, and timezone offsets as specified in [RFC5545] Section 3.3.12 apply.

Example:

```
...
["x-time-local", {}, "time", "12:30:00"],
["x-time-utc", {}, "time", "12:30:00Z"],
["x-time-offset", { "tzid": "Europe/Berlin" }, "time", "12:30:00"],
...
```

3.6.13. URI (RFC5545 section 3.3.13)

Description: iCalendar "URI" property values are represented by a property with the type identifier "uri". The value is a string with the URI.

Example:

```
...
["tzurl", {}, "uri", "http://example.org/tz/Europe-Berlin.ics"],
...
```

3.6.14. UTC Offset (RFC5545 section 3.3.14)

Description: iCalendar "UTC-OFFSET" property values are represented by a property with the type identifier "utc-offset". The value is a string with the same UTC offset value specified by [RFC5545], with the exception that the hour and minute components are separated by a ":" character, for consistency with the [ISO.8601.2004] timezone offset, extended format.

Example:

```
...
["tzoffsetfrom", {}, "utc-offset", "-05:00"],
["tzoffsetto", {}, "utc-offset", "+12:45"],
..
```

3.7. Extensions

iCalendar extension properties and property parameters (those with an "X-" prefix in their name) are handled in the same way as other properties and property parameters: the property is represented by an array, the property parameter represented by an object. The property or parameter name uses the same name as for the iCalendar extension, but in lowercase. For example, the "X-FOO" property in iCalendar turns into the "x-foo" jCal property. See [Section 5](#) for how to deal with default values for unrecognized extension properties or property parameters.

4. Converting from jCal into iCalendar

When converting component, property and property parameter values, the names SHOULD be converted to uppercase. Although iCalendar names are case insensitive, common practice is to keep them all uppercase following the actual definitions in [[RFC5545](#)].

Backslash escaping and line folding MUST be applied to the resulting iCalendar data as required by [[RFC5545](#)].

Non-binary value types MUST NOT be base64 encoded.

jCal properties that do not specify the default type for the iCalendar property MUST add a VALUE parameter when converting to iCalendar. jCal properties that specify the default type SHOULD NOT add a VALUE parameter.

5. Handling Unrecognized Properties or Parameters

In iCalendar, properties have a default value type specified by their definition, e.g. "SUMMARY"'s value type is "TEXT" and "DURATION"'s is "DURATION". When a property uses its default value type, the "VALUE" property parameter does not need to be specified on the property.

When new properties are defined or "X-" properties used, an iCalendar to jCal converter might not recognize them, and not know what the appropriate default value types are, yet they need to be able to preserve the values. A similar issue arises for unrecognized property parameters. As a result, the following rules are applied when dealing with unrecognized properties and property parameters:

- o When converting iCalendar into jCal:

- * Any property that does not include a "VALUE" property parameter and whose default value type is not known, MUST be converted to

a string object. The content of that string is the unprocessed value text.

- * Any unrecognized property parameter MUST be converted to a string value, with its content set to the property parameter value text, treated as if it were a "TEXT" value.
- o When converting jCal into iCalendar:
 - * Since jCal always explicitly specifies the value type, it can always be converted to iCalendar using the VALUE parameter.
 - * If the value type specified in jCal matches the default value type in iCalendar, the VALUE parameter SHOULD be omitted.

Example: The following is an example of an unrecognized iCalendar property (that uses a "DATE-TIME" value as its default), and the equivalent jCal representation of that property.

iCalendar:

```
X-PROPERTY:20110512T120000Z
```

jCal:

```
...  
["x-property", {}, "text", "20110512T120000Z"],  
...
```

Example: The following is an example of a jCal property (where the corresponding iCalendar property uses a "INTEGER" value as its default), and the equivalent iCalendar representation of that property.

jCal:

```
...  
["percent-complete", {}, "integer", 95],  
...
```

iCalendar:

```
PERCENT-COMPLETE:95
```

Example: The following is an example of an unrecognized iCalendar property parameter (that uses a "FLOAT" value as its default) specified on a recognized iCalendar property, and the equivalent jCal representation of that property and property parameter.

iCalendar:

```
DTSTART;X-PARAM=30.3;VALUE=DATE:20110512
```

jCal:

```
...  
["dtstart", { "x-param": "30.3" }, "date", "2011-05-12"],  
...
```

6. Security Considerations

For security considerations specific to calendar data, see [Section 7 of \[RFC5545\]](#). Since this specification is a mapping from iCalendar, no new security concerns are introduced related to calendar data.

The use of JSON as a format does have security risks. [Section 7 of \[RFC4627\]](#) discusses these risks.

7. IANA Considerations

This document defines a MIME media type for use with iCalendar in JSON data. This media type SHOULD be used for the transfer of calendaring data in JSON.

Type name: application

Subtype name: calendar+json

Required parameters: none

Optional parameters: method, component and optinfo as defined for the text/calendar media type in [\[RFC5545\]](#).

Encoding considerations: Same as encoding considerations of application/json as specified in [\[RFC4627\]](#).

Security considerations: See [Section 6](#).

Interoperability considerations: This media type provides an alternative format for iCalendar data based on JSON.

Published specification: This specification.

Applications which use this media type: Applications that currently make use of the text/calendar media type 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.

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

Intended usage: COMMON

Restrictions on usage: There are no restrictions on where this media type can be used.

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

Change controller: IETF

8. Acknowledgments

The authors would like to thank the following for their valuable contributions: William Gill, Erwin Rehme, and Dave Thewlis. This specification originated from the work of the XML-JSON technical committee of the Calendaring and Scheduling Consortium.

9. References

9.1. Normative References

- [ISO.8601.2004] International Organization for Standardization, "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO 8601, 2004.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), January 2005.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 4648](#), October 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](#), January 2008.

- [RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", [RFC 5545](#), September 2009.
- [RFC6321] Daboo, C., Douglass, M., and S. Lees, "xCal: The XML Format for iCalendar", [RFC 6321](#), August 2011.

9.2. Informative References

- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", [RFC 4627](#), July 2006.
- [calconnect-artifacts] The Calendaring and Scheduling Consortium, "Code Artifacts and Schemas", <<http://www.calconnect.org/artifacts.shtml>>.

Appendix A. ABNF Schema

Below is an ABNF schema as per [\[RFC5234\]](#) for iCalendar in JSON. ABNF Symbols not described here are taken from [\[RFC4627\]](#). The schema is non-normative and given for reference only.

The numeric section numbers given in the comments refer to section in [\[RFC5545\]](#). Additional semantic restrictions apply, especially regarding the allowed properties and sub-components per component. Details on these restrictions can be found in this document and [\[RFC5545\]](#).

Additional schemas may be available on the internet at [\[calconnect-artifacts\]](#).

```
; An iCalendar Stream is an array with the first element being the
; string "icalendar". All remaining elements are jcalobjects.
```

```
jcalstream = begin-array
              DQUOTE "icalendar" DQUOTE
              *(value-separator jcalobject)
              end-array
```

```
; A jCal Object is a component with the component-name "vcalendar".
; Restrictions to which properties and sub-components may be
; specified are to be taken from RFC5545.
```

```
jcalobject = component
```

```
; A jCal component consists of the name string, properties array and
; component array
component = begin-array
```

```
    DQUOTE component-name DQUOTE value-separator
    properties-array value-separator
    components-array
    end-array

components-array = begin-array
    [ component *(value-separator component) ]
    end-array

; A jCal property consists of the name string, parameters object,
; type string and one or more values as specified in this document.
property = begin-array
    DQUOTE property-name DQUOTE value-separator
    params-object value-separator
    DQUOTE type-name DQUOTE
    property-value *(value-separator property-value)
    end-array
properties-array = begin-array
    [ property *(value-separator property) ]
    end-array

; Property values depend on the type-name. Aside from the value types
; mentioned here, extensions may make use of other JSON value types.
property-value = string / number / boolean

; The jCal params-object is a JSON object which follows the semantic
; guidelines described in this document.
params-object = begin-object
    [ params-member *(value-separator params-member) ]
    end-object
params-member = DQUOTE param-name DQUOTE name-separator param-value
param-value = string

; The type MUST be a valid type as described by this document. New
; value types can be added by extensions.
type-name = "binary" / "boolean" / "cal-address" / "date" /
    "date-time" / "duration" / "float" / "integer" /
    "period" / "recur" / "text" / "time" / "uri" /
    "utc-offset" / x-type

; Component, property, parameter and type names MUST be lowercase.
; Additional semantic restrictions apply as described by this
; document and RFC5545.
component-name = lowercase-name
property-name = lowercase-name
param-name = lowercase-name
x-type = lowercase-name
```



```
lowercase-name = 1*(%x61-7A / DIGIT / "-")
```

Appendix B. Examples

This section contains two examples of iCalendar objects with their jCal representation.

B.1. Example 1

B.1.1. iCalendar Data

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//Example Inc.//Example Calendar//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:20080205T191224Z
DTSTART:20081006
SUMMARY:Planning meeting
UID:4088E990AD89CB3DBB484909
END:VEVENT
END:VCALENDAR
```

B.1.2. jCal Data

```
[ "vcalendar",
  [
    [ "calscale", {}, "text", "GREGORIAN" ],
    [ "prodid", {}, "text", "-//Example Inc.//Example Calendar//EN" ],
    [ "version", {}, "text", "2.0" ]
  ],
  [
    [ "vevent",
      [
        [ "dtstamp", {}, "date-time", "2008-02-05T19:12:24Z" ],
        [ "dtstart", {}, "date", "2008-10-06" ],
        [ "summary", {}, "text", "Planning meeting" ],
        [ "uid", {}, "text", "4088E990AD89CB3DBB484909" ],
      ],
      []
    ]
  ]
]
```

B.2. Example 2

B.2.1. iCalendar Data

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//Example Client//EN
BEGIN:VTIMEZONE
LAST-MODIFIED:20040110T032845Z
TZID:US/Eastern
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
DTSTAMP:20060206T001121Z
DTSTART;TZID=US/Eastern:20060102T120000
DURATION:PT1H
RRULE:FREQ=DAILY;COUNT=5
RDATE;TZID=US/Eastern;VALUE=PERIOD:20060102T150000/PT2H
SUMMARY:Event #2
DESCRIPTION:We are having a meeting all this week at 12 pm fo
r one hour\, with an additional meeting on the first day 2 h
ours long.\nPlease bring your own lunch for the 12 pm meetin
gs.
UID:00959BC664CA650E933C892C@example.com
END:VEVENT
BEGIN:VEVENT
DTSTAMP:20060206T001121Z
DTSTART;TZID=US/Eastern:20060104T140000
DURATION:PT1H
RECURRENCE-ID;TZID=US/Eastern:20060104T120000
SUMMARY:Event #2 bis
UID:00959BC664CA650E933C892C@example.com
END:VEVENT
END:VCALENDAR
```

B.2.2. jCal Data

```

["vcalendar",
 [
  ["prodid", {}, "text", "-//Example Corp./Example Client//EN"],
  ["version", {}, "text", "2.0"],
 ],
 [
  ["vtimezone",
  [
    ["last-modified", {}, "date-time", "2004-01-10T03:28:45Z"],
    ["tzid", {}, "text", "US/Eastern"],
  ],
  [
    ["daylight",
    [
      ["dtstart", {}, "date-time", "2000-04-04T02:00:00"],
      ["rrule",
      {
        "recur",
        "FREQ=YEARLY;BYDAY=1SU;BYMONTH=4"
      }
      ],
      ["tzname", {}, "text", "EDT"],
      ["tzoffsetfrom", {}, "utc-offset", "-05:00"],
      ["tzoffsetto", {}, "utc-offset", "-04:00"]
    ],
    []
  ],
  ["standard",
  [
    ["dtstart", {}, "date-time", "2000-10-26T02:00:00"],
    ["rrule",
    {
      "recur",
      "FREQ=YEARLY;BYDAY=1SU;BYMONTH=10"
    }
    ],
    ["tzname", {}, "text", "EST"],
    ["tzoffsetfrom", {}, "utc-offset", "-04:00"],
    ["tzoffsetto", {}, "utc-offset", "-05:00"]
  ],
  []
  ]
  ],
 ],
 ["vevent",
 [
  ["dtstamp", {}, "date-time", "2006-02-06T00:11:21Z"],
  ["dtstart",

```

```

    { "tzid": "US/Eastern" },
    "date-time",
    "2006-01-02T12:00:00"
  ],
  [ "duration", {}, "duration", "PT1H" ],
  [ "rrule", {}, "recur", "FREQ=DAILY;COUNT=5" ],
  [ "rdate",
    { "tzid": "US/Eastern" },
    "period",
    "2006-01-02T15:00:00/PT2H"
  ],
  [ "summary", {}, "text", "Event #2" ],
  [ "description",
    {},
    "text",
    // Note that comments and string concatenation are not
    // allowed per JSON specification and is used here only
    // to avoid long lines.
    "We are having a meeting all this week at 12 pm for one " +
    "hour, with an additional meeting on the first day 2 " +
    "hours long.\nPlease bring your own lunch for the 12 pm " +
    "meetings."
  ],
  [ "uid", {}, "text", "00959BC664CA650E933C892C@example.com" ]
],
[]
],
[ "vevent",
  [
    [ "dtstamp", {}, "date-time", "2006-02-06T00:11:21Z" ],
    [ "dtstart",
      { "tzid": "US/Eastern" },
      "date-time",
      "2006-01-02T14:00:00"
    ],
    [ "duration", {}, "duration", "PT1H" ],
    [ "recurrence-id",
      { "tzid": "US/Eastern" },
      "date-time",
      "2006-01-04T12:00:00"
    ],
    [ "summary", {}, "text", "Event #2" ],
    [ "uid", {}, "text", "00959BC664CA650E933C892C@example.com" ]
  ],
  []
]
]
]

```

[Appendix C](#). Change History (to be removed prior to publication as an RFC)

[draft-kewisch-et-al-icalendar-in-json-01](#)

- * Added information on how to handle multi-value parameter. The decision leads to a cleaner draft for a similar proposal for vcard.
- * Removed the open discussion point section regarding the mime media type in favor of adding one.

Authors' Addresses

Philipp Kewisch
Mozilla Corporation
650 Castro Street, Suite 300
Mountain View, CA 94041
USA

EEmail: mozilla@kewis.ch
URI: <http://www.mozilla.org/>

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

EEmail: cyrus@daboo.name
URI: <http://www.apple.com/>

Mike Douglass
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180
USA

EEmail: douglm@rpi.edu
URI: <http://www.rpi.edu/>