JSON data formats for vCard and iCalendar
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2013

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

June 10, 2013

jCal: The JSON format for iCalendar
draft-ietf-jcardcal-jcal-04

Abstract

This specification defines "jCal", a JSON format for iCalendar data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Conventions Used in This Document 3
3. Converting from iCalendar to jCal 4
   3.1. Pre-processing 4
   3.2. iCalendar stream (RFC5545 section 3.4) 5
   3.3. Components (RFC5545 section 3.6) 6
   3.4. Properties (RFC5545 section 3.7 and 3.8) 6
      3.4.1. Special Cases for Properties 7
         3.4.1.1. Multi-valued Properties (RFC5545 Section 3.1.2) . 7
         3.4.1.2. GEO Property (RFC5545 Section 3.8.1.6) 8
         3.4.1.3. REQUEST-STATUS Property (RFC5545 Section 3.8.8.3) 8
   3.5. Parameters (RFC5545 section 3.2) 9
      3.5.1. VALUE parameter 10
      3.5.2. Multi-value Parameters 10
   3.6. Values (RFC5545 section 3.3) 11
      3.6.1. Binary (RFC5545 section 3.3.1) 11
      3.6.2. Boolean (RFC5545 section 3.3.2) 11
      3.6.3. Calendar User Address (RFC5545 section 3.3.3) 12
      3.6.4. Date (RFC5545 section 3.3.4) 12
      3.6.5. Date-Time (RFC5545 section 3.3.5) 12
      3.6.6. Duration (RFC5545 section 3.3.6) 13
      3.6.7. Float (RFC5545 section 3.3.7) 13
      3.6.8. Integer (RFC5545 section 3.3.8) 13
      3.6.9. Period of Time (RFC5545 section 3.3.9) 14
      3.6.10. Recurrence Rule (RFC5545 section 3.3.10) 14
      3.6.11. Text (RFC5545 section 3.3.11) 15
      3.6.12. Time (RFC5545 section 3.3.12) 16
      3.6.13. URI (RFC5545 section 3.3.13) 16
      3.6.14. UTC Offset (RFC5545 section 3.3.14) 17
   3.7. Extensions 17
4. Converting from jCal into iCalendar 17
5. Handling Unrecognized Properties or Parameters 18
6. Implementation Status (to be removed prior to publication as
   an RFC) 20
7. Security Considerations 21
8. IANA Considerations 21
   8.1. UNKNOWN iCalendar Value Data Type 22
9. Acknowledgments 22
10. References 22
   10.1. Normative References 23
   10.2. Informative References 23
Appendix A. ABNF Schema 24
Appendix B. Examples 25
   B.1. Example 1 26
      B.1.1. iCalendar Data 26
      B.1.2. jCal Data 26

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) 30
Authors' Addresses 31

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.

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. The only escaping that may be applied is any escaping mandated by JSON.

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 document.

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:VTODO" and "END:VTODO".

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.

The iCalendar to jCal component mapping is valid for both current iCalendar components and any new iCalendar components added in the future. Conversion is to be done in the same way.

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.

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)

Various iCalendar properties defined in [RFC5545], for example the "CATEGORIES" property, are defined as multi-valued properties. In jCal these properties are added as further members of the array describing the property.

Note that additional multi-valued properties may be added in extensions to the iCalendar format.

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 value is represented as an array of two values. 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.

This kind of special casing can be seen as a "structured value", as described in [I-D.ietf-jcardcal-jcard], Section 3.3.1.3. Implementors may want to keep this in mind in case they aim to support both jCard and jCal in their product. For example, checking the data type of all property values instead of assuming an array for just GEO is RECOMMENDED.

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 value is represented as an array with two or three values. The first array element corresponds to the code, the second element corresponds to the description and the third element 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.

This kind of special casing can be seen as a "structured value", as described in [I-D.ietf-jcardcal-jcard], Section 3.3.1.3.

Implementors may want to keep this in mind in case they aim to support both jCard and jCal in their product. For example, checking the data type of all property values instead of assuming an array for just REQUEST-STATUS is RECOMMENDED.

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"],
       ...
     ],
     ...
   ]
```

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 jCal into iCalendar, the appropriate "VALUE" property parameter MUST be included in the iCalendar property if the value type is not "unknown" or the default value type for that property. See Section 5 for information on handling unknown value types.

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 of a parameter with multiple values 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. An example for a such parameter is the iCalendar "DELEGATED-FROM" and "DELEGATED-TO" parameter, more such parameters may be added in extensions.

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:jdoe@example.org"
    },
    "cal-address",
    "mailto:jsmith@example.org"
   ],
   ...
```

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" properties 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" properties are represented by a property with the type identifier "boolean". 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" properties 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" properties 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" properties 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" properties 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" properties 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" properties 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" properties are represented by a jCal property with the type identifier "period". The property value is an array, with the first element representing the start of the period and the second element representing the end of the period. As in [RFC5545], the start of the period is always formatted as a date-time value and the end of the period MUST be either a date-time or duration value. Any date, date-time or duration values contained in the period value MUST be formatted in accordance to the rules for date, date-time or duration values specified in this document.

Example:

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

3.6.10. Recurrence Rule (RFC5545 section 3.3.10)

Description: iCalendar "RECUR" properties are represented by a property with the type identifier "recur". The value is an object describing the structured data as specified by [RFC5545]. Each rule part is described by the combination of key and value. The key specifies the name of the rule part and MUST be converted to lowercase. The value of the rule part MUST be mapped by the following rules:

* The value of the "freq" and "wskt" rule parts MUST be a string as specified in [RFC5545], with case preserved.

* The value of the "until" rule part MUST be a date or date-time value formatted in accordance to the rules for date or date-time specified in this document.

* The "count" and "interval" rule parts MUST be specified as a single number value.

* The following rule parts can have one or more numeric values: "bysecond", "byminute", "byhour", "bymonthday", "byyearday", "byweekno", "bymonth", and "bysetpos". If a rule part contains multiple values, an array of numbers MUST be used for that rule part. If a rule part contains a single value, the value SHOULD be represented using a single number value, instead of an array with one number value (although an array MAY be used).

* Similarly, the "byday" rule part can have one or more string values. If it contains multiple values, an array of strings MUST be used. If it contains a single value, the value SHOULD be represented using a single string value, instead of an array with one string value (although an array MAY be used).

Example 1:

```
...
["rrule",
 {},
 "recur",
 {
   "freq": "YEARLY",
   "count": 5,
   "byday": [ "-1SU", "2MO" ],
   "bymonth": 10
 }
],
...
```

Example 2:

```
...
["rrule",
 {},
 "recur",
 {
   "freq": "MONTHLY",
   "interval": 2,
   "bymonthday": [ 1, 15, -1 ],
   "until": "2013-10-01"
 }
],
...
```

3.6.11. Text (RFC5545 section 3.3.11)

Description: iCalendar "TEXT" properties 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" properties 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" properties 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" properties 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.

In jCal, a new UNKNOWN property value type is introduced. Its purpose is to allow preserving unknown property values when round-tripping between jCal and iCalendar.

Value name: UNKNOWN

Purpose: To allow preserving unknown property values during round-tripping

Format definition: (Not applicable)

Description: The UNKNOWN value data type is reserved for the exclusive use of the jCal format RFCTODO. It MUST NOT be used in plain iCalendar [RFC5545]. Conversion rules are as follows:

* 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 primitive JSON string. The content of that string is the unprocessed value text. Also, value type MUST be set to "unknown".

  + To correctly implement this format, it is critical that if the default type is not known that the type "unknown" is used. If this requirement is ignored and for example "text" is used, additional escaping may occur which breaks round-tripping values.

  + 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.

* 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.

+ If the value type specified in jCal is set to "unknown", the value MUST be taken over in iCalendar without processing. In this case, the VALUE parameter MUST NOT be specified.

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", {}, "unknown", "20110512T120000Z"],
...
```

Example: The following is an example of how to cope with jCal data where the parser was unable to identify the type. Note how the "unknown" value type is not added to the iCalendar data and escaping, aside from standard JSON string escaping, is not processed.

jCal:

```
...
["x-coffee-data", {}, "unknown", "Stenophylla;Guinea\\,Africa"],
...
```

iCalendar:

```
X-COFFEE-DATA:Stenophylla;Guinea\,Africa
```

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. Implementation Status (to be removed prior to publication as an RFC)

This section describes libraries known to implement this draft as per [I-D.sheffer-running-code].

1. ICAL.js - Philipp Kewisch, James Lal. A JavaScript parser for iCalendar (rfc5545)

   Source: https://github.com/mozilla-comm/ical.js/

   Maturity: production

   Coverage: All aspects of this draft, up to version 01. Includes an online validator. (as of rev 847c67c501, 2013-02-14)

Licensing: MPL, Mozilla Public License 2.0

2. Py Calendar - Cyrus Daboo. iCalendar/vCard Library

Source: https://svn.calendarserver.org/repository/calendarserver/PyCalendar/branches/json/

Maturity: production

Coverage: All aspects of this draft, up to version 01.

Licensing: Apache License, Version 2.0

Additionally, interoperability testing of this draft is an ongoing effort under members of calconnect, the Calendaring and Scheduling Consortium. CalDAV Vendors are looking into supporting this draft.

7. 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.

8. 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 7.

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

8.1. UNKNOWN iCalendar Value Data Type

IANA has added the following entry to the iCalendar Data Types registry, defined in Section 8.3.4 of [RFC5545].

Value Data Type	Status	Reference
UNKNOWN	Current	RFCTODO, Section 5

9. Acknowledgments

The authors would like to thank the following for their valuable contributions: William Gill, Erwin Rehme, and Dave Thewlis, Simon Perreault, Michael Angstadt, Peter Saint-Andre, Bert Greevenbosch, Javier Godoy. This specification originated from the work of the XML-JSON technical committee of the Calendaring and Scheduling Consortium.

10. References

10.1. Normative References

[ISO.8601.2004]
International Organization for Standardization, ""Data elements and interchange formats -- Information interchange -- Representation of dates and times" ", ISO 8601, 12 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.

10.2. Informative References

[I-D.ietf-jcardcal-jcard]
Kewisch, P., "jCard: The JSON format for vCard", draft-ietf-jcardcal-jcard-01 (work in progress), April 2013.

[I-D.sheffer-running-code]
Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: the Implementation Status Section", draft-sheffer-running-code-02 (work in progress), January 2013.

[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
           propery-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.
   ; The non-terminal symbol sructured-prop-value covers the special
   ; cases for GEO and REQUEST-STATUS
   property-value = simple-prop-value / structured-prop-value
   simple-prop-value = string / number / boolean
   structured-prop-value =
       begin-array
       [ structured-element *(value-separator structured-element) ]
       end-array
   structured-element = simple-prop-value

   ; 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 / param-multi
   param-multi = begin-array
                 [ string *(value-separtor string) ]
                 end-array

   ; 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.

* Minor corrections in wording and typo fixes.

draft-kewisch-et-al-icalendar-in-json-02

* Added implementation status section.

* Removed various text tables that just show a conversion from uppercase to lowercase.

* Changed value format for RECUR and PERIOD types

* Minor corrections in wording and typo fixes.

draft-ietf-jcardcal-jcal-00

* Publication as a WG draft

draft-ietf-jcardcal-jcal-01

* Changed handling of GEO and REQUEST-STATUS to align with jCard

* Corrections and additions to the ABNF Section

* Added a further sentence on preprocessing and escaping to clarify that JSON escaping must be used.

draft-ietf-jcardcal-jcal-02

* Changed handling of unknown property parameter types.

* Minor corrections and fixing typos.

draft-ietf-jcardcal-jcal-03

* Changed wording around RECUR value handling.

* Minor corrections and fixing typos.

draft-ietf-jcardcal-jcal-04

* Changed wording around RECUR value handling again.

Authors' Addresses

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

EMail: mozilla@kewis.ch
URI: http://www.mozilla.org/

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

EMail: cyrus@daboo.name
URI: http://www.apple.com/

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

EMail: douglm@rpi.edu
URI: http://www.rpi.edu/