Calendaring Extensions
Internet-Draft
Intended status: Informational
Expires: October 20, 2018

R. Tse
P. Tam
Ribose
C. Daboo
Apple Inc.
K. Murchison
FastMail Pty Ltd
April 18, 2018

# The vObject Model and vFormat Syntax
# draft-calconnect-vobject-vformat-00

Abstract

This document specifies the vObject data model and its corresponding syntax vFormat.

vObject represents the generalized data model, and vFormat the generalized data format, of the following specifications and fully covers them:

- RFC 6350, vCard version 4.0: the VCARD component;
- RFC 5545, Internet Calendaring and Scheduling Core Object Specification (iCalendar): the VCALENDAR, VEVENT, VJOURNAL, VFREEBUSY, VTIMEZONE, VALARM, VTODO, STANDARD and DAYLIGHT components;
- RFC 7953, Calendar Availability Extensions: the VAVAILABILITY and AVAILABLE components;
- I-Ddaboo-icalendar-vpatch, iCalendar Patching: the VPATCH component; and
- alternative formats for iCalendar and vCard, including RFC 6321, xCal; RFC 7265, jCal; RFC 6351, xCard; and RFC 7095, jCard.

This work is produced by the CalConnect TC-VCARD and TC-CALENDAR committees [CALCONNECT-VCARD].

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on October 20, 2018.



Table of Contents

## 1. Introduction

The ubiquitous vCard [RFC6350] and iCalendar [RFC5545] standards are known as the "vObject" or "VCOMPONENT" family of standards. Named by the convention where the component type identifiers usually start with the letter "v", all of them use very similar, if not identical, syntax.

Yet due to diverged implementations of these "vObject" standards, different implementations often generate different output even when given identical content, causing interoperability concerns and the general inability to determine equivalence of vObjects for integrity concerns (Section 2.1.2 of [RFC3552]).

This document defines the "vObject" data model that is a generalized model that fully covers the needs of these standards, and the "vFormat" serialization syntax that is the generalized syntax of all vObject-compliant serialization formats.

This document also describes the normalized form of the vObject data model and the normalization process for vFormat syntax.

This work is produced by the CalConnect TC-VCARD committee [CALCONNECT-VCARD]. The normalized forms and normalization methods described in this document are fully compatible with the vCard 4.0 [RFC6350] and iCalendar [RFC5545] specifications.

## 2. Terms and Definitions

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

The key words "*Private Use*", "*Experimental Use*", "*Hierarchical Allocation*", "*First Come First Served*", "*Expert Review*", "*Specification Required*", "*RFC Required*", "*IETF Review*", "*Standards Action*" and "*IESG Approval*" in this document are to be interpreted as described in Section 4 of [RFC8126].

Notation in this document is described in ABNF [RFC5234] as used by [RFC6350].

Definitions from [RFC6350] apply to this specification except when explicitly overridden.

All names of properties, property parameters, enumerated property values, and property parameter values are case-insensitive. However, all property values are case-sensitive, unless otherwise stated.

### 2.1. Definitions

vObject, VCOMPONENT
: a generalized format of the vCard component (VCARD) and iCalendar (VCALENDAR) component.

vFormat
the textual format representation using the "BEGIN:" and "END:" component keywords

inner vObject, sub-component
a vObject located within a vObject.

outer vObject, super-component
a vObject that this vObject is located within.

Client User Application (CUA)
the vObject client implementation that interfaces with the user

Calendar Date
as defined in [ISO.8601.2004] 2.1.8

## 3. Symbols And Abbreviations

### 3.1. Operators

bitlen(S)
The length of string S in bits (e.g., bitlen(101) == 3).

S + T
addition of two 32-bit vectors S and T with a mod 2^32 bit wrap around.

sort(list)
sorts a list according to alphabetical order (A-Z) of its elements.

upcase(char)
characters in lower case *SHOULD* be replaced with its corresponding upper case character. The input is expected to be in UTF-8 ASCII.

## 4. vObject Data Model

The vObject data model is the generalized data model behind vCard [RFC6350] and iCalendar [RFC5545].

Both vCard [RFC6350] and iCalendar [RFC5545] data formats belong to a family defined here as "vFormat". The "vFormat" format is a generalized data format of both vCard and iCalendar standards, and both these formats adhere to the vFormat requirements.

### 4.1. vObject compliant models

These formats are vObject "compliant formats" as they adhere to the rules of vObject:

- vCard version 4.0 [RFC6350]: the VCARD component
- Internet Calendaring and Scheduling Core Object Specification (iCalendar) [RFC5545], the VCALENDAR, VEVENT, VJOURNAL, VFREEBUSY, VTIMEZONE, VALARM, VTODO, STANDARD, DAYLIGHT components
- Calendar Availability Extensions [RFC7953]: the VAVAILABILITY, AVAILABLE components
- iCalendar Patching [VPATCH]: the VPATCH component

### 4.2. Elements

Data within a vObject is arranged through a hierarchy that is composed of the following elements:

- component;
- property;
- property parameter;
- property value;
- property parameter value; and
- property group.

### 4.3. Equivalence

Two vObjects are considered identical in content if their normalized forms are textually equivalent.

### 4.4. Component

A vObject is based on the representation of the "component" element. All vObjects are composed of at least one component element.

A component:

- *MAY* contain vObject components;
- *MAY* contain properties; and

- *MUST* contain a uniqueness identifier property whose value is called the component's "unique identifier".

A component type is identified by the component name, and every component instance is distinguished by the value of its uniqueness identifier property.

A component is often used to model an object in the object-oriented sense, such as a vCard or an iCalendar object.

The order of components *MAY* represent order of the objects, which is important for example in a "VPATCH" component [VPATCH], representing the patching order, which is not a commutative process.

### 4.4.1. Uniqueness Identifying Property

As a vObject component can contain inner components, it is important that those inner components can be distinguished from each other.

A vObject component's uniqueness identifier property is a property whose value can uniquely identify the immediate component that contains it.

Every vObject component *MUST* contain a single, component uniqueness identifier property. The uniqueness identifier property can be different according to component types, and the uniqueness scope of that property *MAY* be different. At a minimum, the uniqueness identifier property value *MUST* be unique within the immediate component that contains the uniqueness identifier property.

For example:

- a "VCARD" component can be distinguished among other "VCARD" components by its "UID" property value that is globally unique;

- a "STANDARD" component of "VTIMEZONE" can be distinguished according to its "DTSTART" property within the "VTIMEZONE" component that contains it.

The list of uniqueness identifier properties for every vObject component that complies with this document can be found in the IANA registry described in Section 12.1.2.

New vObject components defined according to this document *MUST* define a uniqueness identifier property for that component, and it *MUST* be registered in the said IANA registry.

### 4.4.2. Normalizing A Component

#### 4.4.2.1. Sorted Component Properties

Properties *MUST* be first normalized and before sorted, meaning that the property's values, and its parameters and its values, have been normalized.

The sorting of component properties *MUST* be performed according to the following order:

- alphabetically by the property name. If a property spans multiple content lines, the content lines *MUST NOT* be separated after sorting.
- alphabetically by their normalized value.
- alphabetically by treating its parameters as long strings

#### 4.4.2.2. Sorted Inner Components

Inner components within a vObject must be placed in a sorted order.

The sorting between components *MUST* be performed according to the following order:

- alphabetically by component type, according to component name, such as "VCALENDAR"; then
- if of the same component name, alphabetically according to its unique identifier Section 4.4.1.

#### 4.4.2.3. Normalized Inner Components

An inner component *MUST* itself be normalized, meaning that its properties and inner components *MUST* be normalized.

### 4.4.3. vObject Property

Properties represent attributes of the vObject itself.

A property:

- *MUST* contain a value;
- *MAY* contain one or more property parameters.

vObject property value types are listed in Section 4.4.4.

#### 4.4.3.1. Normalized Values

A normalized property *MUST* have normalized values.

#### 4.4.3.2. Normalized Parameters

A normalized property *MUST* have normalized property parameters.

Property parameters within the same property *MUST* each be normalized, and then sorted by parameter name alphabetically.

Property parameters of the same property *MUST* have unique names.

### 4.4.4. Property Value

A vObject property *MAY* have one or more values, depending on the value type.

vObject property values are strongly typed, just like in [RFC5545] and [RFC6350]. Basic value types accepted in vObject properties are defined in Section 6.

vObject compliant formats *MAY* define additional value types that are not provided in this document, and *MAY* require separate validation rules, such as the "RECUR" property value type from iCalendar [RFC5545].

Each property *MUST* define a default value type, and *MAY* accept alternative, defined, value types.

### 4.4.5. Normalization Of Property Values

The property value generally does not require any normalization. Please consult individual normalization instructions in each value type's definition.

### 4.4.6. Property Parameter

A property parameter is an attribute that applies on a property.

A property parameter contains:

- an identifier identifying its type;
- the value of the property parameter.

A property parameter *MAY* represent:

- information about the property; or

- information about the property value

A property *MAY* have multiple property parameters, for example, the "TYPE" of the value or a category that applies to this value.

#### 4.4.6.1. Value Lists

Certain property parameters allow multiple values. There is no defined order among property parameters in a property parameter list.

In normalized form, values in a value list *MUST* be sorted alphabetically.

### 4.4.7. Default Property Parameter Values

Property parameters are allowed to have a default value.

For example, in vObject formats including [RFC5545] and [RFC6350], each property value has a specified data type specified as the "VALUE" property parameter.

### 4.4.8. Property Parameter Value

A property parameter value **MAY* contain none, one or several property parameter values.

There is generally no order enforced among the list property parameter value list.

vObject property parameter values are strongly typed, just like vObject properties, [RFC5545] and [RFC6350]. Basic value types accepted in vObject property parameter values are defined in Section 6.

vObject compliant formats *MAY* define additional value types that are not provided in this document, and *MAY* require separate validation rules, such as the values accepted by the "FBTYPE" property parameter in iCalendar [RFC5545].

Each property parameter *MUST* define its accepted value type. While a property *MAY* accept multiple alternative value types, a property parameter value *MUST* only accept one value type.

#### 4.4.9. Sorted Parameter Values

Property parameter values within a property parameter *MUST* be sorted by value.

#### 4.4.10. Property Group

A property group (or just "group") is used to group a set of properties, useful to represent cases where a group of properties are closely related to each other.

In the vCard [RFC6350], the group is used to tie multiple properties into being displayed together.

Each property *MUST* only have a maximum of one group.

Groups are not ordered and group names are case-insensitive.

## 5. vFormat Syntax

vFormat (the native vObject format) is the syntax originated from the textual representation of the iCalendar [RFC5545] and vCard [RFC6350] formats. Its syntax is mostly traceable back to the original vCard and iCalendar documents [vCard21].

Parsing and modification operations that work on vFormat *SHOULD* work on all its downstream formats, including iCalendar [RFC5545] and vCard [RFC6350].

This is called the "native" format to distinguish it from alternative representations of vObject data, such as those in XML or in JSON.

### 5.1. ABNF Format Definition

The following ABNF notation defines the vFormat syntax, in accordance with [RFC5234], which also defines CRLF, WSP, DQUOTE, VCHAR, ALPHA, and DIGIT.

A vObject is defined by the following notation:

```
vobjects = 1*vobject

vobject = "BEGIN:" comp-name CRLF
          *contentline
          *vobject
          "END:" comp-name CRLF

comp-name = name
```

```
   prop-name = name

   prop-values = prop-value / prop-list / prop-structured

   prop-value = VALUE-CHAR

   prop-list = prop-value *("," prop-value)
     ; An unordered list containing multiple property values

   prop-structured = prop-value *(";" prop-value)
     ; A structured list that consist of multiple property fields
     ; for multiple property values

   contentline = [group "."] prop-name params ":" prop-values CRLF
     ; Folding and unfolding procedures described in Section 3.2 of
     ; <<RFC6350>> applies:
     ;   * When parsing a content line, folded lines *MUST* first be
     ;     unfolded accordingly.
     ;   * When generating a content line, lines longer than 75
     ;     characters *SHOULD* be folded accordingly.
     ;   * When normalizing a content line, the content line *MUST*
     ;     be folded when the line is longer than 75 characters.

   group = name

   params = *(";" param)

   param = name "=" param-value *("," param-value)
     ; Allowed parameters depend on property name.

   name = 1*(ALPHA / DIGIT / "-")

   param-value = *SAFE-CHAR / DQUOTE *QSAFE-CHAR DQUOTE
   param-values = param-value *("," param-single-value)

   NON-ASCII = UTF8-2 / UTF8-3 / UTF8-4
     ; UTF8-{2,3,4} are defined in <<RFC3629>>
     ; TODO: generalize this to UTF-32

   QSAFE-CHAR = WSP / "!" / %x23-7E / NON-ASCII
     ; Any character except CTLs, DQUOTE

   SAFE-CHAR = WSP / "!" / %x23-39 / %x3C-7E / NON-ASCII
     ; Any character except CTLs, DQUOTE, ";", ":"

   VALUE-CHAR = WSP / VCHAR / NON-ASCII
     ; Any textual character
```

#### 5.1.1. Component

A component:

- begins with line of text that starts with "BEGIN:" following with its component name, ending with a line break;

- ends with a line of text that starts with "END:" following with its component name (matching with the "BEGIN:" line), ending with a line break.

The vCard [RFC6350] and iCalendar [RFC5545] data formats both conform to vFormat, and their syntaxes are considered to be restricted instances of the vObject syntax.

##### 5.1.1.1. Component Name In Uppercase

The component name of a vObject *MUST* be uppercased, for both the "BEGIN:" and "END:" content lines.

Example (even though this is not technically allowed by [RFC6350]):

"BEGIN:vCard"

Should be normalized to:

"BEGIN:VCARD"

##### 5.1.1.2. Order Of Inner Components And Properties

Properties *MUST* be placed before inner components are listed.

##### 5.1.1.3. Maintain Validity

Certain vObject formats places certain restrictions or requirements on property line locations. Normalization procedures *MUST NOT* affect the validity of the normalized vObject.

For example, in the VCARD component [RFC6350], the "VERSION" property line is *REQUIRED* to be placed immediately below the "BEGIN" line.

In this case, when normalizing the VCARD component, the common normalization procedure *MUST* be first applied, and the "VERSION" property line *MUST* be restored to the valid location as required by its specifications [RFC6350].

#### 5.1.2. Property

A property can be represented by one content line (a line that ends with a line break), but can also be "folded" (Section 3.1 of [RFC5545]) to use multiple lines.

A property begins with the property name (e.g., "TEL"), followed by a COLON delimiter and the property's value.

##### 5.1.2.1. Uppercased Property Name

The property name *MUST* be normalized to uppercase letters.

##### 5.1.2.2. Normalized Parameters

The last property parameter of a property *MUST NOT* have a trailing SEMICOLON.

##### 5.1.2.3. Wrapped Content Line

When exporting a normalized property content line, it *MUST* be folded at the character limit when it exceeds 75 characters. Each folded line *MUST* be delimited by the character sequence of a line break and a single white space (CRLF, SPACE (U+0020)). This rule only applies to normalized output.

For example, the original form:

```
NOTE:This is a very long description on a long line that exceeds 75 characters.
```

When exported to normalized output *MUST* give out:

```
NOTE:This is a very long description on a long line that exceeds 75 charac
 ters.
```

#### 5.1.3. Property Value

The property's values are defined as the content after the property name and COLON delimiter, until the end of the unfolded content line.

If a property accepts multiple values, the definition of delimitation is defined in Section 6.

vObject compliant formats that defines additional value types *MAY* require separate validation rules on top of vFormat syntax.

If the property value type of a property value is not the default value type, the "VALUE" parameter *MUST* be present to specify the type of the property value.

vFormat representation of different value types are provided in Section 6.

### 5.2. Normalizing Property Values

The property value generally does not require any normalization.

#### 5.2.1. Property Parameter

Property parameters exist between the property name and the COLON delimiter in a property line.

Each property parameter in a list of property parameters *MUST* be separated by a SEMICOLON character.

The property parameter name and the property parameter value is separated with an equal sign ("=").

##### 5.2.1.1. Multiple Property Parameters

If the property accepts multiple property parameters values, they *MUST* be separated by a SEMICOLON character as a list.

##### 5.2.1.2. Expanded Form Of Property Parameter Value List

When there are multiple instances of a property parameter on the same property, such as in "TYPE=home;TYPE=work", it is considered equivalent to "TYPE=home\,work".

##### 5.2.1.3. Uppercased Property Parameter Names

The property parameter name *MUST* be normalized into uppercase letters in form of a Unicode string ([RFC8259] Section 7).

Property parameter names (together with their values) *MUST* be sorted alphabetically.

Example:

"TEL;VALUE=uri;type=home:tel:+1-888-888-8888"

The normalized form is:

"TEL;TYPE="home";VALUE="uri":tel:+1-888-888-8888"

#### 5.2.1.4. Join Identical Property Parameter Names

If a property parameter occurs more than once within a property, the property parameter is considered to contain a list of property parameter values joined by the parameter separator.

Such instances of property parameters with identical names *MUST* be joined into one instance with its value a sorted list of the property parameter values.

For example, the vCard "TEL" property's "TYPE" parameter [RFC6350] describes that "TYPE=home,work" and "TYPE=work;TYPE=home" are considered equivalent.

Example:

"TEL;TYPE=home;Type=work;VALUE=uri:tel:+1-888-888-8888"

The normalized form is:

"TEL;TYPE="home","work";VALUE=uri:tel:+1-888-888-8888"

### 5.2.2. Express Default Property Value Types

In vObject formats including [RFC5545] and [RFC6350], each property value has a specified data type either as specified by property definition or optionally assigned.

When normalizing a property, the property data value type *MUST* always be specified. If the value type is not explicitly specified, it *MUST* be filled in according to the vObject format.

Example:

"TEL:+1-888-888-8888"

The normalized form is:

"TEL;VALUE="text":+1-888-888-8888"

### 5.2.3. Property Parameter Value

If a property parameter accepts multiple values, these value elements *MUST* be separated by a COMMA (U+002C).

Property parameter value elements that contain the COLON (U+003A), SEMICOLON (U+003B), or COMMA (U+002C) character separators *MUST* be specified as quoted-string text values. Property parameter values

*MUST NOT* contain the DQUOTE (U+0022) character. The DQUOTE character is used as a delimiter for parameter values that contain restricted characters or URI text.

## 5.3. Normalizing Property Parameter Values

### 5.3.1. Wrapped Parameter Values

Property parameter values *MUST* be individually wrapped with the DQUOTE characters. In [RFC6350], the DQUOTE character is used to delimit values within a list that contain restricted characters or URI text and is optional.

Example:

"TEL;TYPE=home,work;VALUE=uri:tel:+1-888-888-8888"

The normalized vFormat output is:

"TEL;TYPE="home","work";VALUE="uri":tel:+1-888-888-8888"

### 5.3.2. Property Group

The syntax of a property group is defined in Section 5.

Property groups *MUST NOT* be removed during normalization. This is contrary to [RFC6350] that allows stripping off groups.

# 6. vObject Value Types

## 6.1. Meta Value Types

### 6.1.1. LIST

Properties and parameters *MAY* specify its value to be an unordered list of values. There is no significance to the order of values in a list.

This construct is called the "LIST". Each value in the LIST *MUST* have the same value type.

In vFormat syntax, values in a list of values *MUST* be separated by a delimiting character. The default delimiter character is the COMMA character. An element within any LIST element *MUST NOT* contain an unescaped delimiter character to prevent breaking of the LIST syntax.

A property *MAY* specify that its LIST elements are delimited by another character other than the COMMA, for example, [RFC5545]'s

"RECUR" property uses the SEMICOLON character. This is allowed when explicitly specified in the defining syntax, given the restriction that any element within does not contain an unescaped delimiter character (i.e., in this case, the SEMICOLON).

#### 6.1.1.1. Syntax

When used to describe a value type, the "LIST(value-type)" function is defined as a list of elements separated by the COMMA character, where each of its elements is of value type "value-type".

When used to describe a value type, the "LIST(value-type)" function is defined as a list of elements separated by the COMMA character, where each of its elements is of value type "value-type".

In the case where a delimiter character for the LIST is not a COMMA, the "LIST(value-type, delimiter)" syntax is used to define the delimiter, where each of its elements is of value type "value-type", and the delimiting character is specified to be "delimiter".

#### 6.1.1.2. Example 1

The "NICKNAME" property of [RFC6350] defines its value to be an unordered list of TEXT.

In vObject notation its value type is defined to be:

LIST(TEXT)

#### 6.1.1.3. Example 2

The "RECUR" property of [RFC5545] defines its value to be an unordered list of ASSIGN, separated by the SEMICOLON character.

In vObject notation its value type is defined to be:

LIST(KEYVALUE, ";")

#### 6.1.1.4. Normalizing a LIST

When normalizing a LIST, each value of it *MUST* be normalized, and the values *MUST* be sorted alphabetically.

For example, values of [RFC5545] RESOURCES, FREEBUSY, EXDATE, RDATE, CATEGORIES, *MUST* be sorted alphabetically when normalized.

### 6.1.2. FIELDSET

Some properties and parameters require values defined in terms of multiple parts.

This construct of multiple structured values is called a "FIELDSET". These structured property values *MUST* have their value parts separated by a SEMICOLON character. Each value in FIELDSET *MUST* have the same value type as defined.

In vFormat syntax, an element within any FIELDSET field *MUST NOT* contain an unescaped the SEMICOLON character to prevent breaking of the FIELDSET syntax.

#### 6.1.2.1. Syntax

When used to describe a value type, the "FIELDSET(field-1-value-type, ...)" function is defined as a structure of fields separated by the SEMICOLON character, where each of its fields is of value type "field-i-value-type", where "i" represents the index of the specific field.

#### 6.1.2.2. Example 1

The "CLIENTPIDMAP" property of [RFC6350] takes a tuple of "INTEGER" and "URI" separated by a SEMICOLON.

The notation in vObject given for its value type would be this indicating that the first value is an INTEGER, while the latter value is a URI:

FIELDSET(INTEGER, URI)

#### 6.1.2.3. Example 2

The "N" property of [RFC6350] defines its value of having 5 values at once, and each of these values are a LIST of TEXT.

The notation in vObject given for its value type would be this indicating that there are 5 fields in this FIELDSET, and each value element of it *MUST* be a LIST of TEXT elements:

FIELDSET(5*LIST(TEXT))

#### 6.1.2.4. Normalizing a FIELDSET

When normalizing a FIELDSET, each value *MUST* have been normalized, but the order of FIELDSET elements *MUST NOT* be rearranged. ==== MAP

A "MAP" serves the function of a key-value table. It is realized using the "LIST" value type with values of the value type "KEYVALUE".

Each value in the MAP *MUST* use the "KEYVALUE" value type.

There is no inherent order of the values within a MAP. Values within its key value pairs *MAY* be of different value types as defined.

In vFormat syntax, as with "LIST", values in a list of values *MUST* be separated by a delimiting character. The default delimiter character is the SEMICOLON character. An element within any "MAP" element *MUST NOT* contain an unescaped delimiter character to prevent breaking of the "MAP" syntax.

#### 6.1.2.5. Syntax

This value type is described using the "MAP(kv_1, kv_2, ...)" function, where each "kv_i" represents a property of the value type KEYVALUE.

In the case where a delimiter character for the MAP is not a SEMICOLON, the "MAP(delimiter, kv_1, kv_2)" syntax is used to define the delimiter, where each of its elements is of value type KEYVALUE, and the delimiting character is specified to be "delimiter".

#### 6.1.2.6. Example

The "RECUR" property of [RFC5545] defines its value to be a MAP, separated by the SEMICOLON character.

In vObject notation its value type is defined to be:

```
MAP(
  KEYVALUE(FREQ, TEXT),
  KEYVALUE(UNTIL, ISO-DATE-COMPLETE / ISO-DATE-TIME-BASIC),
  KEYVALUE(COUNT, INTEGER),
  KEYVALUE(INTERVAL, INTEGER),
  KEYVALUE(BYSECOND, LIST(INTEGER)),
  KEYVALUE(BYMINUTE, LIST(INTEGER)),
  KEYVALUE(BYHOUR, LIST(INTEGER)),
  KEYVALUE(BYDAY, LIST(INTEGER)),
  KEYVALUE(BYMONTHDAY, LIST(INTEGER)),
  KEYVALUE(BYYEARDAY, LIST(INTEGER)),
  KEYVALUE(BYWEEKNO, LIST(INTEGER)),
  KEYVALUE(BYMONTH, LIST(INTEGER)),
  KEYVALUE(BYSETPOS, INTEGER),
  KEYVALUE(WKST, TEXT)
)
```

#### 6.1.2.7. Normalizing a MAP

When normalizing a MAP, each value of it *MUST* be normalized, and the values *MUST* be sorted alphabetically according to its "key".

## 6.2. Basic Value Types

### 6.2.1. TEXT

#### 6.2.1.1. Value Name

TEXT

#### 6.2.1.2. Purpose

This value type defines values that contain free-form, human-readable text.

#### 6.2.1.3. Format Definition

```
text       = *(TSAFE-CHAR / ":" / DQUOTE / ESCAPED-CHAR)
  ; Folded according to description above

ESCAPED-CHAR = ("\\" / "\;" / "\," / "\N" / "\n")
  ; \\ encodes \, \N or \n encodes newline
  ; \; encodes ;, \, encodes ,

TSAFE-CHAR = WSP / %x21 / %x23-2B / %x2D-39 / %x3C-5B /
             %x5D-7E / NON-US-ASCII
  ; Any character except CONTROLs not needed by the current
  ; character set, DQUOTE, ";", ":", "\", ","
```

#### 6.2.1.4. Description

For historic reasons, the COMMA and SEMICOLON characters *MUST* be escaped with a preceding BACKSLASH character (U+005C). The historic usage of the "text" value type accepting multiple values *MUST NOT* be used unless the multiple values are provided through a LIST.

An intentional line break *MUST* be represented by the sequence of "\n" or "\N" (BACKSLASH followed by a LATIN SMALL LETTER N (U+006E) or a LATIN CAPITAL LETTER N (U+004E)).

#### 6.2.1.5. Associated parameters

- Language in which the text is represented can be controlled by the "LANGUAGE" property parameter.

#### 6.2.1.6. Example

This multiple line value for the NOTE value of vCard:

TC VCARD
The Calendaring And Scheduling Consortium
July 20, 2017

requires application of the following formatting rules:

- representing line breaks as "\n"
- line folding with CRLF and the folded-line beginning with a single space
- escaping of the COMMA character

and therefore *SHOULD* be represented as:

```
NOTE:TC VCARD\nThe Calendaring And Scheduling Consortium\nJuly 20\, 2017
```

#### 6.2.1.7. Normalization

Values of the TEXT data type *MUST* convert all line breaks ("\n" or "\N") into the character sequence "\n" (BACKSLASH followed by a LATIN SMALL LETTER N (U+006E)).

### 6.2.2. URI

#### 6.2.2.1. Value Name

URI

#### 6.2.2.2. Purpose

This value type defines values that are represented by data referenced by a uniform resource identifier (URI), the value is what the URI points to, not the URI itself.

#### 6.2.2.3. Format Definition

uri = <As defined in Section 3 of [RFC3986]>

#### 6.2.2.4. Description

When a property parameter value is a URI value type, the URI *MUST* be specified as a quoted-string value.

#### 6.2.2.5. Example

This value for the PHOTO property of vCard:

```
PHOTO:http://www.example.com/pub/photos/jqpublic.gif

PHOTO:
 AQEEBQAwdzELMAkGA1UEBhMCVVMxLDAqBgNVBAoTI05ldHNjYXBlIENvbW11bm
 ljYXRpb25zIENvcnBvcmF0aW9uMRwwGgYDVQQLExNJbmZvcm1hdGlvbiBTeXN0
 <...remainder of base64-encoded data...>
```

#### 6.2.2.6. Normalization

No normalization procedures are needed.  ==== BOOLEAN

#### 6.2.2.7. Value Name

BOOLEAN

#### 6.2.2.8. Purpose

This value type is used to identify properties that contain either a "TRUE" or "FALSE" Boolean value.

#### 6.2.2.9. Format Definition

```
boolean    = "TRUE" / "FALSE"
```

#### 6.2.2.10. Description

Parsing of "TRUE" and "FALSE" values *SHOULD* be case-insensitive, but a writer of such value *MUST* only output of this value type in uppercase.

#### 6.2.2.11. Examples

- TRUE
- false
- True
- FaLSe

#### 6.2.2.12. Normalization

Values of the BOOLEAN data type *MUST* be normalized to uppercase, i.e., the values "TRUE" and "FALSE". ==== INTEGER

#### 6.2.2.13. Value Name

INTEGER

(INTEGER-32 for storing 32-bit integer, INTEGER-64 for storing 64-bit integer)

#### 6.2.2.14. Purpose

Representation of a signed integer value.

#### 6.2.2.15. Format Definition

integer = (["+"] / "-") 1*DIGIT

#### 6.2.2.16. Description

If a preceding sign is not specified, the value is assumed positive "". While the format accepts the optional "" PLUS sign, a writer that conforms to this document *SHOULD* not write the "+" sign for clarity reasons.

The valid ranges for INTEGER-32 and INTEGER-64 are:

- INTEGER-32: -2147483648 (-2^31) to 2147483647 (2^31 - 1)

- INTEGER-64: -9223372036854775807 (-2^63) to 9223372036854775808 (2^63 - 1)

#### 6.2.2.17. Examples

- 1234567890
- -1234567890
- +1234567890
- 432109876

#### 6.2.2.18. Normalization

A positive integer when normalized *MUST* not have the optional "+" sign. ==== FLOAT

#### 6.2.2.19. Value Name

FLOAT

#### 6.2.2.20. Purpose

Representation of a real-number value.

#### 6.2.2.21. Format Definition

float = (["+"] / "-") 1*DIGIT ["." 1*DIGIT]

#### 6.2.2.22. Description

If a preceding sign is not specified, the value is assumed positive "+".

Implementations *MUST* support a precision equal or better than that of the IEEE "binary64" format [IEEE.754.2008].

Scientific notation is disallowed.

#### 6.2.2.23. Examples

- 20.30
- 1000000.0000001
- 1.333

- -3.14

#### 6.2.2.24. Normalization

No normalization procedures are needed.

Trailing zeros, such as "100.10000" *MUST* be kept as it indicates accuracy of the number. ==== LANGUAGE-TAG

#### 6.2.2.25. Value Name

LANGUAGE-TAG

#### 6.2.2.26. Purpose

Representing a language tag, as defined in [RFC5646].

#### 6.2.2.27. Format Definition

Defined in [RFC5646].

#### 6.2.2.28. Description

A single language tag

#### 6.2.2.29. Examples

- de
- en-US
- sr-Cyrl
- zh-yue-HK

#### 6.2.2.30. Normalization

The normalization procedure of the LANGUAGE-TAG data type follows the procedure described in Section 2.1.1 of [RFC5646].

- language codes *MUST* be written in lowercase ('mn' Mongolian)
- script codes *MUST* be in lowercase when the initial letter capitalized ('Cyrl' Cyrillic)
- country codes *MUST* be capitalized ('MN' Mongolia)

As the language tag is comprised of a mixture of these components, [RFC5646] provides a rule that applies this procedure across all language tags:

- All subtags, including extension and private use subtags, *MUST* use lowercase letters.
- Except: two-letter subtags that neither appear at the start of the tag nor occur after singletons *MUST* be in uppercase ("en-CA-x-ca" or "sgn-BE-FR").
- Except: four-letter subtags that neither appear at the start of the tag nor occur after singletons *MUST* be in titlecase ("az-Latn-x-latn").

### 6.2.3. Binary

#### 6.2.3.1. Value Name

BINARY

#### 6.2.3.2. Purpose

This value type defines values that contain inline binary data encoded in characters. For example, an inline "ATTACH" property of an iCalendar object or an inline "PHOTO" property image inside a vCard object.

#### 6.2.3.3. Format Definition

```
binary = *(4b-char) [b-end]
  ; A "BASE64" encoded character string, as defined by [RFC4648].

b-end  = (2b-char "==") / (3b-char "=")

b-char = ALPHA / DIGIT / "+" / "/"
```

#### 6.2.3.4. Description

Property values with this value type *MUST* specify the parameter "ENCODING" with parameter value "BASE64", and the inline binary data *MUST* be character encoded using the "BASE64" encoding method defined in [RFC2045].

#### 6.2.3.5. Example

This value for the NOTE value of vCard:

The following is an example of a "BASE64" encoded binary value data:

```
ATTACH;FMTTYPE=image/vnd.microsoft.icon;ENCODING=BASE64;VALUE
=BINARY:AAABAAEAEBAQAAEABAAoAQAAFgAAACgAAAAQAAAAIAAAAAEABAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAAgIAAAICAgADAwMAA////AAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAMwAAAAAABNEMQAAAAAAkQgAAAAAJEREQgAA
ACECQ0QgEgAAQxQzM0E0AABERCRCREQAADRDJEJEQwAAAhA0QwEQAAAAERE
AAAAAAAREQAAAAAAAkQgAAAAAAAMgAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAA
```

### 6.2.4. KEYVALUE

#### 6.2.4.1. Value Name

"KEYVALUE(key, value-type)"

#### 6.2.4.2. Purpose

Representation of a key-value pair: a key "key" linked to a value of value type "value-type".

#### 6.2.4.3. Format Definition

```
assign-key    = *(TSAFE-CHAR)
assign-value  = prop-values

assign        = assign-key "=" assign-value
```

#### 6.2.4.4. Description

The separation delimiter between the key and value is the EQUAL sign character, "=". The value of "KEYVALUE" *MUST NOT* contain the unescaped EQUAL sign character for the avoidance of doubt.

If the "KEYVALUE" value is accepted within a list, the "key" value must be unique amongst the list.

This value type is also a core component of the "MAP" value type.

#### 6.2.4.5. Examples

- "FREQ=WEEKLY"
- "BYHOUR=3,6,9"
- "BYWEEKNO=MO,TU,WE,TH,FR,SA"

#### 6.2.4.6. Normalization

Its value *MUST* be normalized according to the "value-type" of that value.

### 6.3. Date and Time Value Types

These date and time related value types are based on [ISO.8601.2004] and [ISO.8601.2000].

Multiple of such values can be specified using the comma-separated notation.

#### 6.3.1. ISO-DATE-COMPLETE

##### 6.3.1.1. Value Name

ISO-DATE-COMPLETE

##### 6.3.1.2. Purpose

Representation of a complete calendar date defined in [ISO.8601.2004].

##### 6.3.1.3. Format Definition

```
iso-date              = iso-date-value
iso-date-value        = iso-date-fullyear iso-date-month iso-date-mday
iso-date-fullyear     = 4DIGIT
iso-date-month        = 2DIGIT   ;01-12
iso-date-mday         = 2DIGIT   ;01-28, 01-29, 01-30, 01-31
                                 ;based on month/year
```

##### 6.3.1.4. Description

This value format is based on the "basic format" calendar date (specified in [ISO.8601.2004] Section 4.1.2.2 "Complete representations").

The value *MUST* be represented textually as "YYYYMMDD", with its components "YYYY" a four-digit year, "MM" a two-digit month, and "DD" a two-digit day of the month as described in the definition.

#### 6.3.1.5. Example

The following represents July 1, 1997:

o 19970701

#### 6.3.1.6. Normalization

No normalization procedures are needed. ==== ISO-DATE-FLEX

Representation of a calendar date [ISO.8601.2004] that does not require complete representation.

#### 6.3.1.7. Value Name

ISO-DATE-FLEX

#### 6.3.1.8. Purpose

This value type defines a calendar date format that allows entry of a complete calendar date [ISO.8601.2004], a reduced accuracy date [ISO.8601.2004] and a truncated date [ISO.8601.2004].

#### 6.3.1.9. Format Definition

```
iso-date-flex   = iso-date /
                  iso-date-reduced /
                  iso-date-truncated

iso-date-reduced = iso-date-fullyear / iso-date-year-month
iso-date-year-month = iso-date-fullyear "-" iso-date-month

iso-date-truncated =  iso-date-truncated-month-day /
                      iso-date-truncated-month-only /
                      iso-date-truncated-day-only

iso-date-truncated-month-day  = "--" iso-date-month iso-date-mday
iso-date-truncated-month-only =  "--" iso-date-month
iso-date-truncated-day-only   =  "---" iso-date-mday
```

#### 6.3.1.10. Description

This value format accepts:

- a complete calendar date, specified in [ISO.8601.2004] Section 4.1.2.2 "Complete representations",
- a reduced accuracy calendar date, specified in [ISO.8601.2004] Section 4.1.2.3 "Representations with reduced accuracy", and
- a truncated representation calendar date, specified in [ISO.8601.2000] Section 5.2.1.3 "Truncated representations".

The value can be represented in these ways:

- "YYYYMMDD" Complete representation basic format, specified in [ISO.8601.2004] 4.1.2.2.
- "YYYY-MM" Reduced accuracy representation, specified in [ISO.8601.2004] 4.1.2.3 a).
- "YYYY" Reduced accuracy representation, specified in [ISO.8601.2004] 4.1.2.3 b).
- "--MMDD" Truncated representation for a specific day of a month in the implied year, basic format, specified in [ISO.8601.2000] Section 5.2.1.3 d).
- "--MM" Truncated representation for a specific month in the implied year, basic format, specified in [ISO.8601.2000] Section 5.2.1.3 e).
- "---DD" Truncated representation for a specific day in the implied month, basic format, specified in [ISO.8601.2000] Section 5.2.1.3 f).

Example:

- 20170712
- 2017-07
- 2017
- --0712
- --07

- ---12

#### 6.3.1.11. Normalization

No normalization procedures are needed.

### 6.3.2. ISO-TIME-COMPLETE

#### 6.3.2.1. Value Name

ISO-TIME-COMPLETE

#### 6.3.2.2. Purpose

Representation of a complete time of day with a UTC offset [ISO.8601.2004].

#### 6.3.2.3. Format Definition

```
iso-time = time-hour time-minute time-second
           [iso-time-utc / iso-utc-offset]

iso-time-hour    = 2DIGIT        ;00-23
iso-time-minute  = 2DIGIT        ;00-59
iso-time-second  = 2DIGIT        ;00-60
;The "60" value is used to account for positive "leap" seconds.

iso-time-utc     = "Z"
```

#### 6.3.2.4. Description

This value format accepts a time of day value specified as:

- "hhmmss", the basic format of [ISO.8601.2004] Section 4.2.2.2 "Complete representations".

- "hhmmssZ", the first basic format of [ISO.8601.2004] Section 4.2.4 "UTC of day".

- "hhmmss+/-hhmm", "hhmmss+/-hh", the basic formats of [ISO.8601.2004] Section 4.2.5.2 "Local time and the difference from UTC"

The components mean: "hh" a two-digit, 24-hour of the day (00-23), "mm" a two-digit minute in the hour (00-59), and "ss" a two-digit second in the minute (00-60).

The seconds value of 60 *MUST* only be used to account for positive "leap" seconds. Fractions of a second are not supported by this format.

This value indicates "local time" as specified in [ISO.8601.2004] 2.1.16. To indicate UTC time, a "Z" character *MUST* be appended to the basic format as described in [ISO.8601.2004] Section 4.2.4 "UTC of day". To indicate a UTC offset, the "utc-offset" section *MUST* be specified in accordance with [ISO.8601.2004] Section 4.2.5.2.

The value of "hhmmssZ" *MUST* be used instead of the equivalent "hhmmss+0000" or "hhmmss-0000".

Example:

- 140000
- 140000Z
- 140000-05
- 140000-0500

#### 6.3.2.5. Normalization

No normalization procedures are needed.

### 6.3.3. ISO-TIME-BASIC

#### 6.3.3.1. Value Name

ISO-TIME-BASIC

#### 6.3.3.2. Purpose

Representation of a complete time of day disallowing a UTC offset [ISO.8601.2004].

#### 6.3.3.3. Format Definition

```
iso-time-basic = iso-time-hour iso-time-minute iso-time-second
                 [iso-time-utc]
```

#### 6.3.3.4. Description

This value format is similar to "TIME" except it disallows the additional UTC offset, (the basic formats of [ISO.8601.2004] Section 4.2.5.2 "Local time and the difference from UTC").

This value format accepts a time of day value specified as:

- "hhmmss", the basic format of [ISO.8601.2004] Section 4.2.2.2 "Complete representations".

- "hhmmssZ", the first basic format of [ISO.8601.2004] Section 4.2.4 "UTC of day".

The seconds value of 60 *MUST* only be used to account for positive "leap" seconds. Fractions of a second are not supported by this format.

This value indicates "local time" as specified in [ISO.8601.2004] 2.1.16. To indicate UTC time, a "Z" character *MUST* be appended to the basic format as described in [ISO.8601.2004] Section 4.2.4 "UTC of day".

Example:

- 232050

- 232050Z

#### 6.3.3.5. Normalization

No normalization procedures are needed. ==== ISO-TIME-FLEX

#### 6.3.3.6. Value Name

ISO-TIME-FLEX

#### 6.3.3.7. Purpose

This value type defines a time of day format that allows a entry of a complete time of day [ISO.8601.2004], a reduced accuracy date [ISO.8601.2004] and a truncated date representation [ISO.8601.2000].

#### 6.3.3.8. Format Definition

```
iso-time-flex = iso-time /
                iso-time-reduced /
                iso-time-truncated

iso-time-zone = iso-time-utc / iso-time-utc-offset

iso-time-reduced =  iso-time-reduced-hour-minute /
                    iso-time-hour

iso-time-reduced-hour-minute = iso-time-hour iso-time-minute

iso-time-truncated =  iso-time-truncated-minute-second /
                      iso-time-truncated-minute-only /
                      iso-time-truncated-second-only
iso-time-truncated-minute-second = "-" iso-time-minute iso-time-second
iso-time-truncated-minute-only = "-" iso-time-minute
iso-time-truncated-second-only = "--" iso-time-second
```

#### 6.3.3.9. Description

This value format accepts:

- a complete time of day, specified in [ISO.8601.2004] Section 4.2.2.2 "Complete representations",

- a reduced accuracy time of day, specified in [ISO.8601.2004] Section 4.2.2.3 "Representations with reduced accuracy",

- and a truncated representation time of day, specified in [ISO.8601.2000] Section 5.3.1.4 "Truncated representations".

The value can be represented in these ways:

- "hhmmss" Complete representation basic format, specified in [ISO.8601.2004] 4.2.2.2.

- "hhmm" Reduced accuracy representation basic format for a specific hour and minute, specified in [ISO.8601.2004] 4.2.2.3 a).

- "hh" Reduced accuracy representation basic format for a specific hour, specified in [ISO.8601.2004] 4.2.2.3 b).

- "-mmss" Truncated representation for a specific minute and second of the implied hour, specified in [ISO.8601.2000] Sections 5.3.1.4 a).

- "-mm" Truncated representation for a specific minute of the implied hour, specified in [ISO.8601.2000] Sections 5.3.1.4 b).

- "--ss" Truncated representation for a specific second of the implied minute, specified in [ISO.8601.2000] Sections 5.3.1.4 c).

The seconds value of 60 *MUST* only be used to account for positive "leap" seconds. Fractions of a second are not supported by this format.

This value indicates "local time" as specified in [ISO.8601.2004] 2.1.16. To indicate UTC time, a "Z" character *MUST* be appended to the basic format as described in [ISO.8601.2004] Section 4.2.4 "UTC of day".

This format requires the midnight hour to be represented by "00" ([ISO.8601.2004], Section 4.2.3 a)), not "24" ([ISO.8601.2004], Section 4.2.3 b)).

This format supports the specificiation of UTC offsets for the complete representation basic format (defined in [ISO.8601.2004], Section 4.2.5.2 basic format), in the form of "hhmmss+/-HHMM". "HHMM" is the hour and minute of UTC offset, defined in [ISO.8601.2004], Section 4.2.5.1 basic format.

Example:

- 102200
- 1022
- 10
- -2200
- --00
- 102200Z
- 102200+0800

#### 6.3.3.10. Normalization

No normalization procedures are needed.

### 6.3.4. ISO-UTC-OFFSET

#### 6.3.4.1. Value Name

ISO-UTC-OFFSET

#### 6.3.4.2. Purpose

Representation of a UTC offset as described in [ISO.8601.2004].

#### 6.3.4.3. Format Definition

```
sign = "+" / "-"
iso-utc-offset = sign iso-time-hour [iso-time-minute]
```

Description:

The value can be represented in two ways:

- "+/-hhmm" specified in [ISO.8601.2004] 4.2.5.1 "Difference between local time and UTC of day", the first basic format.

- "+/-hh" specified in [ISO.8601.2004] 4.2.5.1 "Difference between local time and UTC of day", the second basic format.

The PLUS SIGN character MUST be specified for positive UTC offsets (i.e., ahead of UTC). The HYPHEN-MINUS character MUST be specified for negative UTC offsets (i.e., behind of UTC).

The value of "-00" and "-0000" are not allowed. The time-minute, if present, MUST NOT be 60; if absent, it defaults to zero.

#### 6.3.4.4. Example

The following UTC offsets are given for standard time for New York (five hours behind UTC) and Geneva (one hour ahead of UTC):

- -05

- -0500

- +01

- +0100

#### 6.3.4.5. Normalization

No normalization procedures are needed. ==== CAL-UTC-OFFSET

#### 6.3.4.6. Value Name

CAL-UTC-OFFSET

#### 6.3.4.7. Purpose

Representation of a UTC offset as described in [RFC5545].

#### 6.3.4.8. Format Definition

cal-utc-offset = sign iso-time-hour iso-time-minute [iso-time-second]

#### 6.3.4.9. Description:

The value can be represented in two ways:

- "+/-hhmm" specified in [ISO.8601.2004] 4.2.5.1 "Difference between local time and UTC of day", the first basic format.

- "+/-hhmmss" which is unique to this value type.

The PLUS SIGN character MUST be specified for positive UTC offsets (i.e., ahead of UTC). The HYPHEN-MINUS character MUST be specified for negative UTC offsets (i.e., behind of UTC).

The value of "-0000" and "-000000" are not allowed. The time-second, if present, MUST NOT be 60; if absent, it defaults to zero.

#### 6.3.4.10. Example

The following UTC offsets are given for standard time for New York (five hours behind UTC) and Geneva (one hour ahead of UTC):

- -0500

- -050000

- +0100

- +010000

#### 6.3.4.11. Normalization

No normalization procedures are needed. ==== ISO-DATE-TIME-COMPLETE

#### 6.3.4.12. Value Name

ISO-DATE-TIME-COMPLETE

#### 6.3.4.13. Purpose

A complete date and time of day combination as specified in [ISO.8601.2004], Section 4.3.2.

#### 6.3.4.14. Format Definition

iso-date-time = iso-date "T" iso-time

#### 6.3.4.15. Description

This value format accepts a complete date and time of day representation, specified in [ISO.8601.2004] Section 4.3.2 "Complete representations".

The value can be represented in these ways:

- "YYYYMMDDThhmmss" 4.3.2 Complete representation basic format, first entry.
- "YYYYMMDDThhmmssZ" 4.3.2 Complete representation basic format, second entry.
- "YYYYMMDDThhmmss+/-hhmm" 4.3.2 Complete representation basic format, third entry.
- "YYYYMMDDThhmmss+/-hh" 4.3.2 Complete representation basic format, fourth entry.

Examples

- 19850412T101530
- 19850412T101530Z
- 19850412T101530+0400
- 19850412T101530+04

#### 6.3.4.16. Normalization

No normalization procedures are needed. ==== ISO-DATE-TIME-BASIC

#### 6.3.4.17. Value Name

ISO-DATE-TIME-BASIC

#### 6.3.4.18. Purpose

A date and time of day combination without non-UTC timezone as specified in [ISO.8601.2004], Section 4.3.2.

#### 6.3.4.19. Format Definition

iso-date-time-no-zone = iso-date "T" iso-time-basic

#### 6.3.4.20. Description

This value format accepts a complete date and time of day representation, specified in [ISO.8601.2004] Section 4.3.2 "Complete representations", identical with ISO-DATE-TIME-COMPLETE, except that the "utc-offset" portion is disallowed.

The value can be represented in these ways:

- "YYYYMMDDThhmmss" 4.3.2 Complete representation basic format, first entry.

- "YYYYMMDDThhmmssZ" 4.3.2 Complete representation basic format, second entry.

Due to the lack of "utc-offset", properties that use this value type *SHOULD* handle time zone information with other methods such as in property parameters, such as using the "TZID" property parameter defined in [RFC5545].

Example

  * 19980118T230000

  * 19980118T230000Z

#### 6.3.4.21. Normalization

No normalization procedures are needed. ==== ISO-DATE-TIME-FLEX

#### 6.3.4.22. Value Name

DATE-TIME-FLEX

#### 6.3.4.23. Purpose

This value type defines a date and time of day combination as specified in [ISO.8601.2004] Section 4.3 and [ISO.8601.2000], Section 5.4.2 c).

#### 6.3.4.24. Format Definition

```
iso-date-time-flex = iso-date-time-flex-date "T" iso-date-time-flex-time

iso-date-time-flex-date = iso-date / iso-date-truncated
iso-date-time-flex-time = iso-time / iso-time-reduced
```

#### 6.3.4.25. Description

This value format allows for the:

- truncation of the date portion and
- the reduced accuracy of the time portion
- according to the requirements of [ISO.8601.2000], Section 5.4.2 "Representations other than complete" part c).

  Example

- 19961022T150236
- --1022T1502
- ---22T15

#### 6.3.4.26. Normalization

No normalization procedures are needed. ==== ISO-DATE-AND-OR-TIME

#### 6.3.4.27. Value Name

ISO-DATE-AND-OR-TIME

#### 6.3.4.28. Purpose

Representation of a ISO-DATE-FLEX, ISO-TIME-FLEX or an ISO-DATE-TIME-FLEX value.

#### 6.3.4.29. Format Definition

```
iso-date-and-or-time = iso-date-flex /
                       "T" iso-time-flex /
                       iso-date-time-flex
```

#### 6.3.4.30. Description

This value format accepts values of ISO-DATE-FLEX, ISO-TIME-FLEX and ISO-DATE-TIME-FLEX.

A stand-alone ISO-TIME value *MUST* be preceded by a "T" for unambiguous interpretation.

#### 6.3.4.31. Example

- 19961022T140000
- --1022T1400
- ---22T14
- 19850412
- 1985-04
- 1985
- --0412
- ---12
- T102200
- T1022
- T10
- T-2200
- T--00
- T102200Z
- T102200-0800

#### 6.3.4.32. Normalization

No normalization procedures are needed. ==== ISO-DURATION-COMPLETE

#### 6.3.4.33. Value Name

ISO-DURATION-COMPLETE

#### 6.3.4.34. Purpose

Representing a duration of time specified by [ISO.8604.2004] Section 4.4.3.2 complete representation basic format.

#### 6.3.4.35. Format Definition

```
iso-duration-sign = ["+"] / "-"
iso-duration  = ( iso-duration-sign ) "P" iso-duration-value

iso-duration-value =  iso-duration-date / iso-duration-week

iso-duration-date   = iso-duration-day "T" iso-duration-time

iso-duration-week   = 1*DIGIT "W"

iso-duration-year   = 1*DIGIT "Y"
iso-duration-month  = 1*DIGIT "M"
iso-duration-day    = 1*DIGIT "D"

iso-duration-time   = iso-duration-hour iso-duration-minute
                      iso-duration-second

iso-duration-hour   = 1*DIGIT "H"
iso-duration-minute = 1*DIGIT "M"
iso-duration-second = 1*DIGIT "S"
```

Description

The value format is based on the complete representation basic format specified in [ISO.8601.2004] Section 4.4.3.2.

It accepts the following formats ("nn" represents):

- "PnnW" [ISO.8601.2004] Section 4.4.3.2, complete representation, first basic format, for duration in weeks.

- "PnnYnnMnnDTnnHnnMnnS" [ISO.8601.2004] Section 4.4.3.2, complete representation, second basic format, for duration in years, months, days, hours, minutes and seconds.

This format differs from the specification of [ISO.8601.2004] Section 4.4.3.2 in the following areas:

- An optional, preceding "sign", element is used to indicate positive or negative duration. Negative durations are useful in representing reverse scheduling, such as the time to trigger an alarm before an associated time (see [RFC5545]).

- Reduced accuracy as defined in [ISO.8601.2004] Section 4.4.3.2 is not allowed. Omission of the number and corresponding designator of days, hours, minutes or seconds is not allowed even if any of the expressions are zero ([ISO.8601.2004] Section 4.4.3.2 c)).

- The duration of a week or a day depends on its position in the calendar.

In the case of discontinuities in the time scale, such as the change from standard time to daylight time and back, the computation of the exact duration requires the subtraction or addition of the change of duration of the discontinuity. Leap seconds *MUST NOT* be considered when computing an exact duration.

#### 6.3.4.36. Example

A duration of 15 days, 5 hours, and 20 seconds *MAY* be represented as

- P0Y0M15DT5H0M20S

A duration of 7 weeks *MAY* be represented as:

- P7W

#### 6.3.4.37. Normalization

No normalization procedures are needed.

### 6.3.5. CAL-DURATION

#### 6.3.5.1. Value Name

CAL-DURATION

#### 6.3.5.2. Purpose

Representing a duration of time specified by [ISO.8604.2004] Section 4.4.3.2 complete representation basic format, simiar to ISO-DURATION, but with syntax tailored to calendaring.

#### 6.3.5.3. Format Definition

```
cal-duration         = cal-duration-sign cal-duration-no-sign
cal-duration-sign    = (["+"] / "-")
cal-duration-no-sign = "P" cal-duration-value

cal-duration-value   = ( cal-duration-date /
                         cal-duration-time /
                         cal-duration-week )

cal-duration-date    = cal-duration-day [cal-duration-time]

cal-duration-time    = "T" ( cal-duration-hour /
                             cal-duration-minute /
                             cal-duration-second )

cal-duration-week    = 1*DIGIT "W"
cal-duration-hour    = 1*DIGIT "H" [cal-duration-minute]
cal-duration-minute  = 1*DIGIT "M" [cal-duration-second]
cal-duration-second  = 1*DIGIT "S"
cal-duration-day     = 1*DIGIT "D"
```

#### 6.3.5.4. Description

The value format is similar to ISO-DURATION and based on the complete representation basic format specified in [ISO.8601.2004] Section 4.4.3.2, but given extra flexibility to calendaring usage.

It accepts the following formats ("nn" represents):

- "PnnW" [ISO.8601.2004] Section 4.4.3.2, complete representation, first basic format, for duration in weeks.

- "PnnDTnnHnnMnnS" [ISO.8601.2004] Section 4.4.3.2, complete representation, second basic format, with the omission of years and months, for duration in days, hours, minutes and seconds.

- "PnnDTnnHnnM" Reduced accuracy with omission of seconds.

- "PnnDTnnH" Reduced accuracy with omission of minutes.

- "PnnD" Reduced accuracy with omission of hours.

This format differs from the specification of [ISO.8601.2004] Section 4.4.3.2 in the following areas:

- Years and months are not accepted in this syntax.

- An optional, preceding "sign", element is used to indicate positive or negative duration. Negative durations are useful in representing reverse scheduling, such as the time to trigger an alarm before an associated time (see [RFC5545]).

- Reduced accuracy is allowed for in particular, the omission of the number and designators of hours, minutes or seconds is allowed with the omission starting from the extreme right-hand side. In the case of the omission of the time value, the "T" separator *MUST* also be omitted. The day ("D") portion *MUST* always be present.

- The duration of a week or a day depends on its position in the calendar.

In the case of discontinuities in the time scale, such as the change from standard time to daylight time and back, the computation of the exact duration requires the subtraction or addition of the change of duration of the discontinuity. Leap seconds *MUST NOT* be considered when computing an exact duration.

When computing an exact duration, the greatest order time components *MUST* be added first, that is, the number of days *MUST* be added first, followed by the number of hours, number of minutes, and number of seconds.

#### 6.3.5.5. Example

A duration of 0 days, 0 hours, and 20 seconds *SHOULD* be represented as

P0DT0H0M20S

A duration of 15 days, 5 hours, and 3 hours *SHOULD* be represented as

P15DT5H3M

A duration of 15 days, 5 hours *SHOULD* be represented as

P15DT5H

A duration of 15 days *SHOULD* be represented as

P15D

A duration of 7 weeks *SHOULD* be represented as:

P7W

#### 6.3.5.6. Normalization

No normalization procedures are needed. ==== ISO-INTERVAL

#### 6.3.5.7. Value Name

ISO-INTERVAL-COMPLETE

#### 6.3.5.8. Purpose

Representation of a time interval.

#### 6.3.5.9. Format Definition

```
iso-interval     = iso-interval-explicit / iso-interval-start

iso-interval-explicit = iso-date-time "/" iso-date-time
iso-interval-start    = iso-date-time "/" iso-duration-no-sign
```

#### 6.3.5.10. Description

This value format accepts a time interval representation, specified in [ISO.8601.2004] Section 4.4. "Time Interval".

The value can be represented by:

a) a start and an end;

- "YYYYMMDDThhmmss/YYYYMMDDThhmmss" [ISO.8601.2004] 4.4.4.1 Complete representation, "Representations of time intervals identified by start and end", basic format, first entry. The start *MUST* be before the end.

c) a start and a duration;

- "YYYYMMDDThhmmss/PnnYnnMnnDTnnHnnMnnS" [ISO.8601.2004] 4.4.4.3 Complete representation, "Representations of time interval identified by start and duration", first basic format. The duration component *MUST* be positive.

- "YYYYMMDDThhmmss/PnnW" [ISO.8601.2004] 4.4.4.5 Other complete representations, third item, allowing the expression "PnnYnnMnnDTnnHnnMnnS" to be substituted with "PnnW" [ISO.8601.2004] 4.4.3.2.

d) a duration and an end.

- "PnnYnnMnnDTnnHnnMnnS/YYYYMMDDThhmmss" [ISO.8601.2004] 4.4.4.4 Complete representation, "Representations of time interval identified by duration and end", first basic format. The start of the interval can be determined by subtracting the duration component from the end of the interval.

- "PnnW/YYYYMMDDThhmmss" [ISO.8601.2004] 4.4.4.5 Other complete representations, third item, allowing the expression "PnnYnnMnnDTnnHnnMnnS" to be substituted with "PnnW" [ISO.8601.2004] 4.4.3.2.

In accordance with [ISO.8601.2004] Section 4.4.4.5:

- where representations using local time in a time point component are shown, a complete representation of UTC ([ISO.8601.2004] Section 4.2.4) or local time and the difference from UTC ([ISO.8601.2004] Section 4.2.5.2) *MAY* be substituted for local time, i.e. representations using the expression "YYYYMMDDThhmmss" could be substituted with any of these:

- "YYYYMMDDThhmmssZ" 4.3.2 Complete representation basic format, second entry.

- "YYYYMMDDThhmmss+/-hhmm" 4.3.2 Complete representation basic format, third entry.

- "YYYYMMDDThhmmss+/-hh" [ISO.8601.2004] 4.3.2 Complete representation basic format, fourth entry.

In accordance with [ISO.8601.2004] Section 4.4.5:

- representations for UTC included with the component preceding the solidus shall be assumed to apply to the component following the solidus, unless a corresponding alternative is included.

  Example

- 19850412T232050/P1Y2M15DT12H30M0S

- 19850412T232050Z/P1Y2M15DT12H30M0S

- 19850412T232050Z/19850612T232050

- P1Y2M15DT12H30M0S/19850412T232050

#### 6.3.5.11. Normalization

No normalization procedures are needed.

### 6.3.6. CAL-INTERVAL

#### 6.3.6.1. Value Name

CAL-INTERVAL

#### 6.3.6.2. Purpose

Representation of a time interval for calendaring.

#### 6.3.6.3. Format Definition

```
cal-interval     = cal-interval-explicit / cal-interval-start

cal-interval-explicit = iso-date-time-no-zone "/" iso-date-time-no-zone
cal-interval-start    = iso-date-time-no-zone "/" cal-duration-no-sign
```

Description
This value format accepts a time interval representation, specified in [ISO.8601.2004] Section 4.4. "Time Interval" tailored for calendaring purposes.

The value can be represented in two ways.

As an interval with start and end:

- "YYYYMMDDThhmmss/YYYYMMDDThhmmss" [ISO.8601.2004] 4.4.4.1 Complete representation, "Representations of time intervals identified by start and end", basic format, first entry. The start *MUST* be before the end.

As an interval with start and duration (positive duration only):

- "YYYYMMDDThhmmss/PnnDTnnHnnMnnS" [ISO.8601.2004] 4.4.4.3 Complete representation, "Representations of time interval identified by start and duration", first basic format, modified to omit the "nnYnnM", which is the "cal-duration" period format.

- "YYYYMMDDThhmmss/PnnW" [ISO.8601.2004] 4.4.4.5 Other complete representations, third item, allowing the expression "PnnYnnMnnDTnnHnnMnnS" to be substituted with "PnnW" [ISO.8601.2004] 4.4.3.2.

- "YYYYMMDDThhmmss/PnnDTnnHnnM" with the duration specified in reduced accuracy with omission of seconds as in (#cal-duration).

- "YYYYMMDDThhmmss/PnnDTnnH" with the duration specified in reduced accuracy with omission of minutes as in (#cal-duration).

- "YYYYMMDDThhmmss/PnnD" with the duration specified in reduced accuracy with omission of hours as in (#cal-duration).

In accordance with [ISO.8601.2004] Section 4.4.5, representations for UTC included with the component preceding the solidus shall be assumed to apply to the component following the solidus, unless a corresponding alternative is included.

Example

- 19970101T180000Z/19970102T070000Z

- 19850412T232050/19850625T103000

- 19970101T180000Z/PT5H30M

- 19850412T232050/P15DT12H30M0S

- 19850412T232050/P00010215T123000

- Both components are in UTC: 19850412T232050Z/19850625T103000

- Former component in local time, latter in UTC: 19850412T232050/19850625T103000

#### 6.3.6.4. Normalization

No normalization procedures are needed.

### 6.4. Parameter Value Types

TODO: vFormat URI in parameter values should be wrapped with DQUOTES.

#### 6.4.1. PARAM-VALUE

##### 6.4.1.1. Value Name

PARAM-VALUE

#### 6.4.1.2. Purpose

This parameter value type is the basic value type for parameter values.

#### 6.4.1.3. Format Definition

```
param-value   = paramtext / quoted-string

paramtext     = *SAFE-CHAR
quoted-string = DQUOTE *QSAFE-CHAR DQUOTE

QSAFE-CHAR    = WSP / %x21 / %x23-7E / NON-US-ASCII
    ; Any character except CONTROL and DQUOTE

SAFE-CHAR     = WSP / %x21 / %x23-2B / %x2D-39 / %x3C-7E
              / NON-US-ASCII
    ; Any character except CONTROL, DQUOTE, ";", ":", ","

NON-US-ASCII  = UTF8-2 / UTF8-3 / UTF8-4
     ; UTF8-2, UTF8-3, and UTF8-4 are defined in [RFC3629]

CONTROL       = %x00-08 / %x0A-1F / %x7F
     ; All the controls except HTAB
```

#### 6.4.1.4. Description

Values that contain the COLON, SEMICOLON, or COMMA character separators *MUST* be specified as quoted-string text values.

Property parameter values that contain the DQUOTE character *MUST* be escaped and specified as quoted-string text values.

Property parameter values that are not in quoted-strings are case-insensitive.

An intentional line break *MUST* be represented by the sequence of "\n" or "\N" (BACKSLASH followed by a LATIN SMALL LETTER N (U+006E) or a LATIN CAPITAL LETTER N (U+004E)).

#### 6.4.1.5. Example

The value "cid:part1.0001@example.org" in the parameter "ALTREP" within a "DESCRIPTION" property in iCalendar can be specified like this:

```
DESCRIPTION;ALTREP="cid:part1.0001@example.org":The Fall'98 Wild
 Wizards Conference - - Las Vegas\, NV\, USA
```

#### 6.4.1.6. Normalization

Values of the PARAM-VALUE parameter data type *MUST* convert all line breaks ("\n" or "\N") into the character sequence "\n" (BACKSLASH followed by a LATIN SMALL LETTER N (U+006E)).

If a value of the PARAM-VALUE type is specified as "paramtext" (i.e., not inside a quoted-string), the value should be down-cased.

### 6.4.2. Conversion Of Property Value Types

All vObject property value types are allowed in parameter values.

Two primitives are defined to convert property value types into an acceptable parameter value type within vFormat:

- "DISALLOW-DQUOTES": converts a vObject property value type to reject DQUOTES within its value

- "WRAP-DQUOTES": converts a vObject property value type to require outermost DQUOTES at the beginning and at the end of its value

## 7. Normalization

A normalization procedure can be applied to vObjects (in its various representations) to make them compatible prior to comparison, allowing for consistent results. The result of normalization processing of a vObject, is an equivalent vObject described according to vFormat representation.

The normalization method has the following properties:

- stable across different implementations generating the same output from the same input

- compatible with alternative representation formats such as xCard [RFC6351] / jCard [RFC7095] and xCal [RFC6321] / jCal [RFC7265]

- generates output adhering to the original vObject format allowing interoperability with existing implementations

- generates output compatible with protocols that utilize these vObject, such as CardDAV [RFC6352] and CalDAV [RFC4791] systems.

There are two levels of normalization.

- vObject normalization, of values and property parameter values, are performed within the vObject data model;

- vFormat normalization, of the format syntax itself, is performed during serialization of a vObject into vFormat.

## 7.1. Approach

The goals of the normalization procedure are:

- A normalized vObject will be a valid vObject in vFormat syntax. Therefore the normalization procedure requires knowledge of the source specific vObject format.

- A normalized vObject is stable across alternative representation formats, such as xCal and jCal of iCalendar, and xCard and jCard of vCard. This allows comparison of vObject content regardless of the representation format.

- Allows comparison of equivalence of content rather than formatting. E.g., addition of new-lines within a vCard and order of listed properties do not affect the resulting normalized form.

- A normalized vObject *MUST* maintain validity under the original format rules, such as in the case of VCARD [RFC6350] components, the "VERSION" property line *MUST* be located immediately after the "BEGIN" property line.

## 7.2. Steps

In order to serialize a vObject into normalized vFormat syntax, one would directly serialize the vObject data model into vFormat syntax.

The steps are generally described below.

1. Normalize the vObject

    A. Normalize properties

        i. Normalize property parameters

            A. Normalize property parameter types

            B. Normalize property parameter values

                I. Sort property parameter values alphabetically.

                II. Concatenate property parameter values.

C. Normalize property parameter key: cast to uppercase.

D. Concatenate string form of property parameter key, value type and values.

ii. Normalize property values

B. Normalize inner vObjects (sub-components)

i. Perform the same function as (1)

### 7.3. Alternative vCard representations

The normalization procedure applies to alternative vObject representations as well, including:

- xCard [RFC6351]
- jCard [RFC7095]
- xCal [RFC6321]
- jCal [RFC7265]

To normalize a vObject provided in these representations, the vObject data should be first normalized in data model form according to Section 4, and then serialized into these representations.

## 8. Client Implementations Recommendations

A CUA *SHOULD* normalize the vObject upon modification of it.

## 9. CardDAV

### 9.1. Additional Server Semantics for PUT, COPY and MOVE

This specification creates an additional precondition and postcondition for the PUT, COPY, and MOVE methods when:

- A PUT operation requests an address object resource to be placed into an address book collection; and
- A COPY or MOVE operation requests an address object resource to be placed into (or out of) an address book collection.

#### 9.1.1. Provide Normalized Output

Certain servers perform silent changes or cleanups of client provided vCard data when stored as address object resources, such as the order of property parameters or scrubbed values.

The resulting vCard data stored on the server (and when returned back to the client) *MAY* end up different than that of the client without its knowledge. It is therefore necessary for the client to be reported on such modifications.

Additional Postcondition:

(CARDDAV:resource-normalized): Convert to normalized format.

## 10. CalDAV

TODO.

## 11. Security Considerations

Security considerations around vObject formats in the following documents *MUST* be adhered to:

- vCard: [RFC6350]
- iCalendar: [RFC5545], [RFC5789], [RFC4791]

## 12. IANA Considerations

New vObject and vFormat specifications produced *MUST* adhere to the requirements, including the normalization process, described in this document, and any exceptions or further instructions for normalization *MUST* be described.

### 12.1. vObject Component Uniqueness Properties Registration Procedure

This section defines the process to register new or modified uniqueness properties for vObject components with IANA.

The IETF will create a mailing list, vobject@ietf.org, which can be used for public discussion of vObject elements prior to registration.

The registry policy is *Specification Required*; any newly proposed specification *MUST* be reviewed by the designated expert.

The registry *SHOULD* contain the following note:

Note: Experts are to verify that the proposed registration *SHOULD* provide benefits for the wider vObject community, and provides a publicly-available standard that can be implemented in an interoperable way. References to IETF-published documents are preferred. The "Reference" value should point to a document that details the implementation of this property.

The registration procedure begins when a completed registration template, defined in the sections below, is sent to vobject@ietf.org and iana@iana.org.

The designated expert is expected to tell IANA and the submitter of the registration within two weeks whether the registration is approved, approved with minor changes, or rejected with cause. When a registration is rejected with cause, it can be re-submitted if the concerns listed in the cause are addressed. Decisions made by the designated expert can be appealed to the IESG Applications Area Director, then to the IESG. They follow the normal appeals procedure for IESG decisions.

### 12.1.1. Registration Template for vObject Component Uniqueness Properties

A registration for a vObject Component Uniqueness Property is defined by completing the following template.

Component
: The name of the component.

Property
: The property of the component that is used to uniquely identify the component it belongs to.

Scope
: The uniqueness scope of the aforementioned property.

Reference
: The document that defines the component syntax and the uniqueness identifying property. Generally, this is where the component was originally defined, but if the uniqueness property is defined in an extension document, a reference to the extension document *SHOULD* be given instead.

Description
: Any special notes about the usage of the uniqueness identifying property, how it is to be used, etc.

Example(s)

One or more examples of instances of the component need to be specified.

### 12.1.2. Initial vObject Component Uniqueness Identifiers Registry

The IANA created and maintains this registry for vObject Component Uniqueness Identifiers with pointers to appropriate reference documents.

The following table has been used to initialize the registry.

| Component | Property | Scope | Reference |
|---|---|---|---|
| VCALENDAR | UID | Global | [RFC7986], Section 5.3 |
| VCARD | UID | Global | [RFC6350], Section 6.7.6 |
| VEVENT | UID | Global | [RFC5545], Section 3.6.1 |
| VTODO | UID | Global | [RFC5545], Section 3.6.2 |
| VJOURNAL | UID | Global | [RFC5545], Section 3.6.3 |
| VFREEBUSY | UID | Global | [RFC5545], Section 3.6.4 |
| VTIMEZONE | TZID | Global | [RFC5545], Section 3.6.5 |
| STANDARD | DTSTART | Parent | [RFC5545], Section 3.6.5 |
| DAYLIGHT | DTSTART | Parent | [RFC5545], Section 3.6.5 |
| VALARM | UID | Global | [I-D.daboo-valarm-extensions], Section 4 |
| VAVAILABILITY | UID | Global | [RFC7953], Section 3.1 |
| AVAILABLE | UID | Global | [RFC7953], Section 3.1 |
| VPOLL | UID | Parent | [I-D.york-vpoll], Section 4.5.1 |
| VVOTER | VOTER | Parent | [I-D.york-vpoll], Section 4.5.2 |
| VOTE | POLL-ITEM-ID | Parent | [I-D.york-vpoll], Section 4.5.3 |

## 13. Mapping Of Data Value Types For Existing RFCs

### 13.1. RFC 6350

| Data Type | Original Data Type |
|---|---|
| BOOLEAN | BOOLEAN |
| ISO-DATE-FLEX | DATE |
| ISO-DATE-AND-OR-TIME | DATE-AND-OR-TIME |
| ISO-DATE-TIME-FLEX | DATE-TIME |
| FLOAT | FLOAT |
| INTEGER-64 | INTEGER |
| LANGUAGE-TAG | LANGUAGE-TAG |
| TEXT | TEXT |
| ISO-TIME-FLEX | TIME |
| ISO-TIME-COMPLETE | TIMESTAMP |
| URI | URI |
| ISO-UTC-OFFSET | UTC-OFFSET |

## 13.2. RFC 5545

| Data Type | Original Data Type |
|---|---|
| BINARY | BINARY |
| BOOLEAN | BOOLEAN |
| URI | CAL-ADDRESS |
| ISO-DATE-COMPLETE | DATE |
| ISO-DATE-TIME-BASIC | DATE-TIME |
| CAL-DURATION | DURATION |
| FLOAT | FLOAT |
| INTEGER-32 | INTEGER |
| CAL-DURATION | PERIOD |
| TEXT | TEXT |
| ISO-TIME-BASIC | TIME |
| URI | URI |
| CAL-UTC-OFFSET | UTC-OFFSET |
| RECURMAP Section 13.2.1 | RECUR |

### 13.2.1. RECURMAP

RECURMAP is shown here instead of within the tables due to space constraints.

It is defined to be:

```
RECURMAP = MAP(
  KEYVALUE(FREQ, TEXT),
  KEYVALUE(UNTIL, ISO-DATE-COMPLETE / ISO-DATE-TIME-BASIC),
  KEYVALUE(COUNT, INTEGER),
  KEYVALUE(INTERVAL, INTEGER),
  KEYVALUE(BYSECOND, LIST(INTEGER)),
  KEYVALUE(BYMINUTE, LIST(INTEGER)),
  KEYVALUE(BYHOUR, LIST(INTEGER)),
  KEYVALUE(BYDAY, LIST(INTEGER)),
  KEYVALUE(BYMONTHDAY, LIST(INTEGER)),
  KEYVALUE(BYYEARDAY, LIST(INTEGER)),
  KEYVALUE(BYWEEKNO, LIST(INTEGER)),
  KEYVALUE(BYMONTH, LIST(INTEGER)),
  KEYVALUE(BYSETPOS, INTEGER),
  KEYVALUE(WKST, TEXT)
)
```

## 14. Mapping Of Component Property Value Types For Existing RFCs

### 14.1. VCARD Component (RFC 6350)

Properties with the default data type as TEXT.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| BEGIN | TEXT | | 1*TEXT |
| END | TEXT | | 1*TEXT |
| KIND | TEXT | | 1*TEXT |
| XML | TEXT | | 1*TEXT |
| FN | TEXT | | 1*TEXT |
| BDAY | ISO-DATE-AND-OR-TIME | TEXT | 1*date-and-or-time, 1*text |
| ANNIVERSARY | ISO-DATE-AND-OR-TIME | TEXT | 1*date-and-or-time, 1*text |
| EMAIL | TEXT | | 1*TEXT, |
| TZ | TEXT | URI, ISO-UTC-OFFSET | 1*TEXT, URI, UTC-OFFSET |
| TITLE | TEXT | | 1*TEXT |
| ROLE | TEXT | | 1*TEXT |
| NOTE | TEXT | | 1*TEXT |
| PRODID | TEXT | | 1*TEXT |
| VERSION | TEXT | | 1*TEXT |

Properties with the default data type as URI.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| TEL | URI | TEXT | 1*text, URI |
| SOURCE | URI | | URI |
| PHOTO | URI | | URI |
| IMPP | URI | | URI |
| GEO | URI | | URI |
| LOGO | URI | | URI |
| MEMBER | URI | | URI |
| RELATED | URI | TEXT | URI, 1*text |
| UID | URI | | URI, 1*text |
| KEY | URI | | URI, 1*text |
| SOUND | URI | | URI |
| URL | URI | | URI |
| FBURL | URI | | URI |
| CALADRURI | URI | | URI |
| CALURI | URI | | URI |

Properties with FIELDSET.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| N | FIELDSET(5*LIST(TEXT)) | | TEXT |
| GENDER | FIELDSET(2*TEXT) | | TEXT |
| ADR | FIELDSET(7*LIST(TEXT)) | | TEXT |
| ORG | FIELDSET(1*TEXT) | | TEXT |
| CLIENTPIDMAP | FIELDSET(INTEGER-64, URI) | | TEXT |

Properties with LIST.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| NICKNAME | LIST(TEXT) | | TEXT |
| CATEGORIES | LIST(TEXT) | | TEXT |

Properties with ISO-DATE-AND-OR-TIME.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| BDAY | ISO-DATE-AND-OR-TIME | TEXT | date-and-or-time, text |
| ANNIVERSARY | ISO-DATE-AND-OR-TIME | TEXT | date-and-or-time, text |

Properties with ISO-DATE-TIME-COMPLETE.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| REV | ISO-DATE-TIME-COMPLETE | | timestamp |

Properties with LANGUAGE-TAG.

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| LANG | LANGUAGE-TAG | | language-tag |

## 14.2. VCALENDAR Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| PRODID | TEXT | | 1*TEXT |
| VERSION | TEXT | | 1*TEXT |
| CALSCALE | TEXT | | 1*TEXT |
| METHOD | TEXT | | 1*TEXT |
| IANA-REGed/X- | TEXT | | 1*TEXT |

## 14.3. VEVENT Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|

| | | | |
|---|---|---|---|
| DTSTAMP | ISO-DATE-TIME-BASIC | | DATE-TIME |
| UID | TEXT | | 1*TEXT |
| DTSTART | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| CLASS | TEXT | | 1*TEXT |
| CREATED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| DESCRIPTION | TEXT | | 1*TEXT |
| GEO | FIELDSET(2*FLOAT) | | FLOAT ";" FLOAT |
| LAST-MODIFIED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| LOCATION | TEXT | | 1*TEXT |
| ORGANIZER | URI | | cal-address |
| PRIORITY | INTEGER-32 | | INTEGER |
| SEQUENCE | INTEGER-32 | | INTEGER |
| STATUS | TEXT | | 1*TEXT |
| SUMMARY | TEXT | | 1*TEXT |
| TRANSP | TEXT | | 1*TEXT |
| URL | URI | | URI |
| RECURRENCE-ID | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| RRULE | RECURMAP Section 13.2.1 | | RECUR |
| DTEND | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| DURATION | DURATION | | DURATION |
| ATTACH | URI | BINARY | URI, BINARY |
| ATTENDEE | URI | | cal-address |
| CATEGORIES | LIST(TEXT) | | TEXT |
| COMMENT | TEXT | | 1*TEXT |
| CONTACT | TEXT | | 1*TEXT |
| EXDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE ) | | DATE-TIME, DATE |
| RELATED-TO | TEXT | | 1*TEXT |
| RESOURCES | LIST(TEXT) | | TEXT |
| RDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE / CAL-INTERVAL) | | DATE-TIME, DATE, PERIOD |
| IANA-REGed/X- | TEXT | | 1*TEXT |

### 14.4. VTODO Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| DTSTAMP | ISO-DATE-TIME-BASIC | | DATE-TIME |
| UID | TEXT | | 1*TEXT |
| CLASS | TEXT | | 1*TEXT |
| CREATED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| COMPLETED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| DESCRIPTION | TEXT | | 1*TEXT |
| DTSTART | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| GEO | FIELDSET(2*FLOAT ) | | FLOAT ";" FLOAT |
| LAST-MODIFIED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| LOCATION | TEXT | | 1*TEXT |
| ORGANIZER | URI | | cal-address |
| PRIORITY | INTEGER-32 | | INTEGER |
| SEQUENCE | INTEGER-32 | | INTEGER |
| STATUS | TEXT | | 1*TEXT |
| SUMMARY | TEXT | | 1*TEXT |
| URL | URI | | URI |
| RRULE | RECURMAP Section 13.2.1 | | RECUR |
| DUE | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| DURATION | DURATION | | DURATION |
| ATTACH | URI | BINARY | URI, BINARY |
| ATTENDEE | URI | | cal-address |
| CATEGORIES | LIST(TEXT) | | TEXT |
| COMMENT | TEXT | | 1*TEXT |
| CONTACT | TEXT | | 1*TEXT |
| EXDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE ) | | DATE-TIME, DATE |
| REQUEST-STATUS | TEXT | | 1*TEXT |
| RELATED-TO | TEXT | | 1*TEXT |
| RESOURCES | LIST(TEXT) | | TEXT |
| RDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE / | | DATE-TIME, DATE, PERIOD |

| | | | | |
|---|---|---|---|---|
| | CAL-INTERVAL) | | | |
| IANA-REGed/X- | TEXT | | 1*TEXT | |

## 14.5. VJOURNAL Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| DTSTAMP | ISO-DATE-TIME-BASIC | | DATE-TIME | |
| UID | TEXT | | 1*TEXT |
| CLASS | TEXT | | 1*TEXT |
| CREATED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| DTSTART | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| LAST-MODIFIED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| ORGANIZER | URI | | cal-address |
| SEQUENCE | INTEGER-32 | | INTEGER |
| STATUS | TEXT | | 1*TEXT |
| SUMMARY | TEXT | | 1*TEXT |
| URL | URI | | URI |
| RRULE | RECURMAP Section 13.2.1 | | RECUR |
| ATTACH | URI | BINARY | URI, BINARY |
| ATTENDEE | URI | | cal-address |
| CATEGORIES | LIST(TEXT) | | TEXT |
| COMMENT | TEXT | | 1*TEXT |
| CONTACT | TEXT | | 1*TEXT |
| DESCRIPTION | TEXT | | 1*TEXT |
| EXDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE ) | | DATE-TIME, DATE |
| RELATED-TO | TEXT | | 1*TEXT |
| RDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE / CAL-INTERVAL) | | DATE-TIME, DATE, PERIOD |
| REQUEST-STATUS | TEXT | | 1*TEXT |
| IANA-REGed/X- | TEXT | | 1*TEXT |

### 14.6. VFREEBUSY Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| DTSTAMP | ISO-DATE-TIME-BASIC | | DATE-TIME |
| UID | TEXT | | 1*TEXT |
| CONTACT | TEXT | | 1*TEXT |
| DTSTART | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| DTEND | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| ORGANIZER | URI | | cal-address |
| URL | URI | | URI |
| ATTENDEE | URI | | cal-address |
| COMMENT | TEXT | | 1*TEXT |
| FREEBUSY | LIST(CAL-INTERVAL) | | LIST(PERIOD) |
| REQUEST-STATUS | TEXT | | 1*TEXT |
| IANA-REGed/X- | TEXT | | 1*TEXT |

### 14.7. VTIMEZONE Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| TZID | TEXT | | 1*TEXT |
| LAST-MODIFIED | ISO-DATE-TIME-BASIC | | DATE-TIME |
| TZURL | URI | | URI |
| IANA-REGed/X- | TEXT | | 1*TEXT |

### 14.8. STANDARD / DAYLIGHT Components (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| DTSTART | ISO-DATE-TIME-BASIC | ISO-DATE-COMPLETE | DATE-TIME, DATE |
| TZOFFSETFROM | CAL-UTC-OFFSET | | UTC-OFFSET |
| TZOFFSETTO | CAL-UTC-OFFSET | | UTC-OFFSET |
| RRULE | RECURMAP Section 13.2.1 | | RECUR |
| COMMENT | TEXT | | 1*TEXT |
| RDATE | LIST( ISO-DATE-TIME-BASIC / ISO-DATE-COMPLETE / CAL-INTERVAL) | | DATE-TIME, DATE, PERIOD |
| TZNAME | TEXT | | 1*TEXT |
| IANA-REGed/X- | TEXT | | 1*TEXT |

## 14.9. VALARM Component (RFC 5545)

| Property | Default Data Type | Alt. Data Types | Original Data Type |
|---|---|---|---|
| ACTION | TEXT | | 1*TEXT |
| DESCRIPTION | TEXT | | 1*TEXT |
| SUMMARY | TEXT | | 1*TEXT |
| TRIGGER | DURATION | ISO-DATE-TIME-BASIC | DURATION, DATE-TIME |
| DURATION | DURATION | | DURATION |
| REPEAT | INTEGER-32 | | INTEGER |
| ATTACH | URI | BINARY | URI, BINARY |
| ATTENDEE | URI | | cal-address |
| IANA-REGed/X- | TEXT | | 1*TEXT |

# 15. Mapping Of Parameter Value Types For Existing RFCs

## 15.1. RFC 6350

| Parameter | Data Type |
|---|---|
| LANGUAGE | LANGUAGE-TAG |
| VALUE | TEXT |
| PREF | INTEGER-64 |
| ALTID | TEXT |
| PID | TEXT |
| TYPE | LIST(TEXT) |
| MEDIATYPE | TEXT |
| CALSCALE | TEXT |
| SORT-AS | LIST(TEXT) |
| GEO | URI |
| TZ | TEXT |

### 15.2. RFC 5545

| Parameter | Data Type |
|---|---|
| ALTREP | URI |
| CN | TEXT |
| CUTYPE | TEXT |
| DELEGATED-FROM | URI |
| DELEGATED-TO | URI |
| DIR | URI |
| ENCODING | TEXT |
| FMTTYPE | TEXT |
| FBTYPE | TEXT |
| LANGUAGE | LANGUAGE-TAG |
| MEMBER | LIST(URI) |
| PARTSTAT | TEXT |
| RANGE | TEXT |
| RELATED | TEXT |
| RELTYPE | TEXT |
| ROLE | TEXT |
| RSVP | BOOLEAN |
| SENT-BY | URI |
| TZID | TEXT |
| VALUE | TEXT |

## 16. References

## 16.1. 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>.

[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>.

[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>.

[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>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[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>.

## 16.2. Informative References

[CALCONNECT-CALENDAR]
The Calendaring and Scheduling Consortium, "CalConnect CALENDAR Technical Committee", April 2018, <https://www.calconnect.org/about/technical-committees/calendar-technical-committee>.

[CALCONNECT-VCARD]
The Calendaring and Scheduling Consortium, "CalConnect VCARD Technical Committee", April 2018, <http://www.calconnect.org/about/technical-committees/vcard-technical-committee>.

[I-D.daboo-valarm-extensions]
Daboo, C., "VALARM Extensions for iCalendar", draft-daboo-valarm-extensions-04 (work in progress), June 2012.

[I-D.york-vpoll]
York, E., Daboo, C., and M. Douglass, "VPOLL: Consensus Scheduling Component for iCalendar", draft-york-vpoll-04 (work in progress), February 2017.

[IEEE.754.2008]
Institute of Electrical and Electronics Engineers, "IEEE Standard 754, Standard for Binary Floating-Point Arithmetic", August 2008, <https://doi.org/10.1109/IEEESTD.2008.4610935>.

[ISO.8601.2000]
ISO/IEC, "ISO 8601:2000, Data elements and interchange formats -- Information interchange -- Representation of dates and times", December 2000, <https://www.iso.org/standard/26780.html>.

[ISO.8601.2004]
ISO/IEC, "ISO 8601:2004, Data elements and interchange formats -- Information interchange -- Representation of dates and times", December 2004, <https://www.iso.org/standard/40874.html>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[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>.

[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>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/info/rfc5789>.

[RFC6321] Daboo, C., Douglass, M., and S. Lees, "xCal: The XML Format for iCalendar", RFC 6321, DOI 10.17487/RFC6321, August 2011, <https://www.rfc-editor.org/info/rfc6321>.

[RFC6351] Perreault, S., "xCard: vCard XML Representation", RFC 6351, DOI 10.17487/RFC6351, August 2011, <https://www.rfc-editor.org/info/rfc6351>.

[RFC6352] Daboo, C., "CardDAV: vCard Extensions to Web Distributed Authoring and Versioning (WebDAV)", RFC 6352, DOI 10.17487/RFC6352, August 2011, <https://www.rfc-editor.org/info/rfc6352>.

[RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <https://www.rfc-editor.org/info/rfc7095>.

[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>.

[RFC7953] Daboo, C. and M. Douglass, "Calendar Availability", RFC 7953, DOI 10.17487/RFC7953, August 2016, <https://www.rfc-editor.org/info/rfc7953>.

[RFC7986] Daboo, C., "New Properties for iCalendar", RFC 7986, DOI 10.17487/RFC7986, October 2016, <https://www.rfc-editor.org/info/rfc7986>.

[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>.

[vCard21] Internet Mail Consortium, "vCard - The Electronic Business Card Version 2.1", 09 1996.

[VPATCH] Daboo, C. and K. Murchison, "The iCalendar VPATCH Component (draft)", 10 2016.

## Appendix A. Normalization Examples for vFormat

Original:

```
BEGIN:VOBJECT
PROPERTY1:10
PROPERTY2:20
END:VOBJECT
```

Normalized:

```
BEGIN:VOBJECT
PROPERTY1:10
PROPERTY2:20
END:VOBJECT
```

### A.1. vCard

Original:

```
BEGIN:VCARD
VERSION:4.0
KIND:individual
FN:Martin Van Buren
N:Van Buren;Martin;;;Hon.
TEL;VALUE=uri;PREF=1;TYPE="voice";TYPE="home":tel:+1-888-888-8888;ext=8888
END:VCARD
```

Normalized:

```
BEGIN:VCARD
VERSION:4.0
KIND:individual
FN:Martin Van Buren
N:Van Buren;Martin;;;Hon.
TEL;VALUE=uri;PREF=1;TYPE="voice,home":tel:+1-888-888-8888;ext=8888
END:VCARD
```

## Appendix B. Acknowledgements

The authors wish to thank their families and the following parties who helped this materialize and for their support of a better and vObject-enabled world:

- CalConnect TC-VCARD committee
- Marten Gajda
- members and the Board of Directors of CalConnect

This specification was developed by the CalConnect TC-VCARD committee.

## Authors' Addresses

Ronald Henry Tse
Ribose
Suite 1111, 1 Pedder Street
Central
Hong Kong

Email: ronald.tse@ribose.com
URI: https://www.ribose.com

Peter Kwan Yu Tam
Ribose
Suite 1111, 1 Pedder Street
Central
Hong Kong

Email: peter.tam@ribose.com
URI: https://www.ribose.com

Cyrus Daboo
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
United States of America

Email: cyrus@daboo.name
URI: https://www.apple.com

Kenneth Murchison
FastMail Pty Ltd
Level 2, 114 William Street
Melbourne, VIC 3000
Australia

Email: murch@fastmailteam.com