Network Working Group
Internet-Draft
Obsoletes: 2425, 2426, 4770
(if approved)
Updates: 2739 (if approved)
Intended status: Standards Track
Expires: November 7, 2009

S. Perreault
Viagenie
P. Resnick
QUALCOMM Incorporated

May 6, 2009

vCard Format Specification
draft-ietf-vcarddav-vcardrev-07

Status of This Memo

Copyright Notice

Abstract

This document defines the vCard data format for representing and exchanging a variety of information about an individual (e.g., formatted and structured name and delivery addresses, email address, multiple telephone numbers, photograph, logo, audio clips, etc.).

Table of Contents

1. Introduction

Note: This draft contains much of the same text as 2425 and 2426 which may not be correct. Those two RFCs have been merged and the structure of this draft is what's new. Some vCard-specific suggestions have been added, but for the most part this is still very open. But we'd like to get feedback on the structure mostly so that it may be fixed.

Electronic address books have become ubiquitous. Their increased presense on portable, connected devices as well as the diversity of platforms exchanging contact data call for a standard. This memo defines the vCard format, which allows the capture and exchange of information normally stored within an address book or directory application.

2. Conventions

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

3. MIME Type Registration

To: ietf-types@iana.org

Subject: Registration of media type text/vcard

Type name: text

Subtype name: vcard

Required parameters: none

Optional parameters: charset

Encoding considerations: The "charset" MIME parameter is interpreted as defined in [RFC2046], section 4.1.2. If it is omitted, the default encoding is UTF-8 as defined in [RFC3629].

Security considerations: See Section 10.

Interoperability considerations: The text/vcard media type is intended to identify vCard data of any version. There are older specifications of vCard [RFC2426][oldreference_VCARD] still in common use. While these formats are similar, they are not strictly compatible. In general, it is necessary to inspect the value of the VERSION property (see Section 7.7.10) for identifying

the standard to which a given vCard object conforms.

In addition, the following media types are known to have been used to refer to vCard data. They should be considered deprecated in favor of text/vcard.

* text/directory

* text/directory; type=vcard

* text/x-vcard

Published specification: draft-ietf-vcarddav-vcardrev-07

Applications that use this media type: They are numerous, diverse, and include mail user agents, instant messaging clients, address book applications, directory servers, customer relationship management software, etc.

Additional information:

Magic number(s):

File extension(s): .vcf

Macintosh file type code(s):

Person & email address to contact for further information: Simon Perreault <simon.perreault@viagenie.ca>

Intended usage: COMMON

Restrictions on usage: none

Author: Simon Perreault and Pete Resnick

Change controller: IETF

4. vCard Format Specification

The text/vcard MIME content type (hereafter known as "vCard") contains contact information, typically pertaining to a single contact or group of contacts. The content consists of one or more lines in the format given below.

4.1. Line Delimiting and Folding

Individual lines within vCard are delimited by the [RFC5322] line break, which is a CRLF sequence (ASCII decimal 13, followed by ASCII decimal 10). Long logical lines of text can be split into a multiple-physical-line representation using the following folding technique. Content lines SHOULD be folded to a maximum width of 75 octets. Multi-octet characters MUST remain contiguous. The rationale for this folding process can be found in [RFC5322], Section 2.1.1.

A logical line MAY be continued on the next physical line anywhere between two characters by inserting a CRLF immediately followed by a single white space character (space, ASCII decimal 32, or horizontal tab, ASCII decimal 9). At least one character must be present on the folded line. Any sequence of CRLF followed immediately by a single white space character is ignored (removed) when processing the content type. For example the line:

```
DESCRIPTION:This is a long description that exists on a long line.
```

can be represented as:

```
DESCRIPTION:This is a long description
  that exists on a long line.
```

It could also be represented as:

```
DESCRIPTION:This is a long descrip
 tion that exists o
 n a long line.
```

The process of moving from this folded multiple-line representation of a property definition to its single line representation is called unfolding. Unfolding is accomplished by regarding CRLF immediately followed by a white space character (namely HTAB ASCII decimal 9 or SPACE ASCII decimal 32) as equivalent to no characters at all (i.e., the CRLF and single white space character are removed).

> Note: It is possible for very simple implementations to generate improperly folded lines in the middle of a UTF-8 multi-octet sequence. For this reason, implementations SHOULD unfold lines in such a way as to properly restore the original sequence.
>
> Note: Unfolding is done differently than in [RFC5322]. Unfolding in [RFC5322] only removes the CRLF, not the space following it.

Folding is done after any content encoding of a type value.

Unfolding is done before any decoding of a type value in a content line.

4.2. ABNF Format Definition

The following ABNF uses the notation of [RFC5234], which also defines CRLF, WSP, DQUOTE, VCHAR, ALPHA, and DIGIT.

```
  vcard-entity = 1*(vcard)

  vcard = "BEGIN" ":" "VCARD" CRLF
          1*contentline
          "END" ":" "VCARD" CRLF
    ;A vCard object MUST include the VERSION and FN properties.

  contentline = [group "."] name *(";" param) ":" value CRLF
    ; When parsing a content line, folded lines MUST first
    ; be unfolded according to the unfolding procedure
    ; described above.
    ; When generating a content line, lines longer than 75
    ; characters SHOULD be folded according to the folding
    ; procedure described above.

  group = "WORK" / "HOME" / iana-token / x-name
  name  = "SOURCE" / "NAME" / "KIND" / "FN" / "N" / "NICKNAME"
        / "PHOTO" / "BDAY" / "DDAY" / "BIRTH" / "DEATH" / "GENDER"
        / "ADR" / "LABEL" / "TEL" / "EMAIL" / "IMPP" / "LANG"
        / "TZ" / "GEO" / "TITLE" / "ROLE" / "LOGO" / "ORG" / "MEMBER"
        / "RELATED" / "CATEGORIES" / "NOTE" / "PRODID" / "REV"
        / "SORT-STRING" / "SOUND" / "UID" / "CLIENTPIDMAP" / "URL"
        / "VERSION" / "CLASS" / "KEY" / "FBURL" / "CALADRURI"
        / "CALURI" / iana-token / x-name
    ; Parsing of the param and value is based on the "name" as
    ; defined in ABNF sections below.
    ; Group and name are case-insensitive.

  iana-token = 1*(ALPHA / DIGIT / "-")
    ; identifier registered with IANA

  x-name = "x-" 1*(ALPHA / DIGIT / "-")
    ; Names that begin with "x-" or "X-" are
    ; reserved for experimental use, not intended for released
    ; products, or for use in bilateral agreements.

  param = language-param / encoding-param / value-param / pref-param
        / pid-param / type-param / geo-param / tz-param / any-param
    ; Allowed parameters depend on property name.
```

```
param-value = *SAFE-CHAR / DQUOTE *QSAFE-CHAR DQUOTE

any-param  = (iana-token / x-name) "=" param-value

NON-ASCII = %x80-FF
  ; Use is restricted by charset parameter
  ; on outer MIME object (UTF-8 by default)

QSAFE-CHAR = WSP / %x21 / %x23-7E / NON-ASCII
  ; Any character except CTLs, DQUOTE

SAFE-CHAR = WSP / %x21 / %x23-39 / %x3C-7E / NON-ASCII
  ; Any character except CTLs, DQUOTE, ";", ":"

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

A line that begins with a white space character is a continuation of the previous line, as described above. The white space character and immediately preceeding CRLF should be discarded when reconstructing the original line. Note that this line-folding convention differs from that found in [RFC5322], in that the sequence <CRLF><WSP> found anywhere in the content indicates a continued line and should be removed.

Property names and parameter names are case insensitive (e.g., the property name "fn" is the same as "FN" and "Fn"). Parameter values MAY be case sensitive or case insensitive, depending on their definition.

The group construct is used to group related properties together. For example, groups named "WORK" and "HOME" could be used to segregate properties such as telephone number, address, etc. Displaying of groups is left entirely up to the application. Predefined groups with assigned meaning are listed in Section 11.3.1. It is possible to register new groups from IANA. Unregistered groups MAY be used and MUST start with "X-".

Properties defined in a vCard instance may have multiple values depending on the property cardinality. The general rule for encoding multi-valued properties is to simply create a new content line for each value (including the property name). However, it should be noted that some value types support encoding multiple values in a single content line by separating the values with a comma ",". This approach has been taken for several of the content types defined below (date, time, integer, float), for space-saving reasons.

4.3. Property Value Escaping

A property instance may contain one or more values delimited by a COMMA character (ASCII decimal 44). Therefore, a COMMA character in a value MUST be escaped with a BACKSLASH character (ASCII decimal 92), even for properties that don't allow multiple instances (for consistency).

Some properties (e.g. N and ADR) comprise multiple fields delimited by a SEMI-COLON character (ASCII decimal 59). Therefore, a SEMI-COLON in a field of such a "compound" property MUST be escaped with a BACKSLASH character. SEMI-COLON characters in non-compound properties MUST NOT be escaped.

Furthermore, some fields of compound properties may contain a list of values delimited by a COMMA character. Therefore, a COMMA character in one of a field's values MUST be escaped with a BACKSLASH character, even for fields that don't allow multiple values (for consistency). Compound properties allowing multiple instances MUST NOT be encoded in a single content line.

Finally, newline (ASCII decimal 10) and backslash (ASCII decimal 92) characters in values MUST be escaped by prefixing them with a backslash character.

In all other cases, escaping MUST NOT be used.

5. Property Value Data Types

Standard value types are defined below.

```
  value = text
        / text-list
        / date-list
        / time-list
        / date-time-list
        / timestamp-list
        / boolean
        / integer-list
        / float-list
        / binary
        / utc-offset
        / URI               ; from Appendix A of [RFC3986]
        / iana-valuespec
    ; Actual value type depends on property name and VALUE parameter.

  text = *VALUE-CHAR
```

```
text-list = *TEXT-LIST-CHAR *("," *TEXT-LIST-CHAR)

TEXT-LIST-CHAR = "\\" / "\," / "\n"
               / <any VALUE-CHAR except , or \ or newline>
   ; Backslashes, commas, and newlines must be encoded.

date-list      = date      *("," date)
time-list      = time      *("," time)
date-time-list = date-time *("," date-time)
timestamp-list = timestamp *("," timestamp)
integer-list   = integer   *("," integer)
float-list     = float     *("," float)

boolean = "TRUE" / "FALSE"
integer = [sign] 1*DIGIT
float   = [sign] 1*DIGIT ["." 1*DIGIT]

sign = "+" / "-"

binary = <A binary string encoded according
          to the "encoding" parameter>

year   = 4DIGIT  ; 0000-9999
month  = 2DIGIT  ; 01-12
day    = 2DIGIT  ; 01-28/29/30/31 depending on month and leap year
hour   = 2DIGIT  ; 00-23
minute = 2DIGIT  ; 00-59
second = 2DIGIT  ; 00-58/59/60 depending on leap second
zone   = "Z" / utc-offset

date          = year     month  day
              / year "-" month
              / "--"     month [day]
              / "--"      "-"   day
date-noreduc  = year     month  day
              / "--"     month  day
              / "--"      "-"   day
date-complete = year     month  day

time          = hour [minute [second]] [zone]
              /  "-"  minute [second]  [zone]
              /  "-"   "-"    second   [zone]
time-notrunc  = hour [minute [second]] [zone]
time-complete = hour  minute  second   [zone]

date-time = date-noreduc  "T" time-notrunc
timestamp = date-complete "T" time-complete
```

```
utc-offset = sign hour [minute]

iana-valuespec = <a publicly-defined valuetype format, registered
                 with IANA, as defined in section 12 of this
                 document>
```

5.1. TEXT

"text": The "text" value type should be used to identify values that contain human-readable text. The character set in which the text is represented is controlled by the "charset" MIME type parameter. Note that there is no way to override this parameter on a per-property basis. As for the language, it is controlled by the "language" property parameter defined in Section 6.1.

Examples for "text":

```
this is a text value
this is one value,this is another
this is a single value\, with a comma encoded
```

A formatted text line break in a text value type MUST be represented as the character sequence backslash (ASCII decimal 92) followed by a Latin small letter n (ASCII decimal 110) or a Latin capital letter N (ASCII decimal 78), that is "\n" or "\N".

For example a multiple line DESCRIPTION value of:

```
Mythical Manager
Hyjinx Software Division
BabsCo, Inc.
```

could be represented as:

```
DESCRIPTION:Mythical Manager\nHyjinx Software Division\n
 BabsCo\, Inc.\n
```

demonstrating the \n literal formatted line break technique, the CRLF-followed-by-space line folding technique, and the backslash escape technique.

5.2. URI

"uri": The "uri" value type should be used to identify values that are referenced by a URI (including a Content-ID URI), instead of encoded in-line. These value references might be used if the value is too large, or otherwise undesirable to include directly. The format for the URI is as defined in [RFC3986]. Note that the value

of a property of type "uri" is what the URI points to, not the URI itself.

Examples for "uri":

```
http://www.example.com/my/picture.jpg
ldap://ldap.example.com/cn=babs%20jensen
```

5.3. DATE, TIME, DATE-TIME, and TIMESTAMP

"date", "time", "date-time", and "timestamp": Each of these value types is based on a the definitions in [ISO.8601.2004] standard. Multiple such values can be specified using the comma-separated notation.

Only the basic format is supported.

5.3.1. DATE

A calendar date as specified in [ISO.8601.2004] section 4.1.2.

Reduced accuracy, as specified in [ISO.8601.2004] sections 4.1.2.3 a) and b), but not c), is permitted.

Expanded representation, as specified in [ISO.8601.2004] section 4.1.4, is forbidden.

Truncated representation, as specified in [ISO.8601.2000] sections 5.2.1.3 d), e), and f), is permitted.

Examples for "date":

```
19850412
1985-04
1985
--0412
---12
```

5.3.2. TIME

A time of day as specified in [ISO.8601.2004] section 4.2.

Reduced accuracy, as specified in [ISO.8601.2004] section 4.2.2.3, is permitted.

Representation with decimal fraction, as specified in [ISO.8601.2004] section 4.2.2.4, is forbidden.

Midnight is always represented by 00, never 24 (see [ISO.8601.2004] section 4.2.3).

Truncated representation, as specified in [ISO.8601.2000] sections 5.3.1.4 a), b), and c), is permitted.

Examples for "time":

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

5.3.3. DATE-TIME

A date and time of day combination as specified in [ISO.8601.2004] section 4.3.

Truncation of the date part, as specified in [ISO.8601.2000] section 5.4.2 c), is permitted.

Examples for "date-time":

```
19961022T140000
--1022T1400
---22T14
```

5.3.4. TIMESTAMP

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

Examples for "timestamp":

```
19961022T140000
19961022T140000Z
19961022T140000-05
19961022T140000-0500
```

5.4. BOOLEAN

"boolean": The "boolean" value type is used to express boolen values. These values are case insensitive.

Examples:

```
TRUE
false
True
```

5.5. INTEGER

"integer": The "integer" value type is used to express signed integers in decimal format. If sign is not specified, the value is assumed positive "+". Multiple "integer" values can be specified using the comma-separated notation.

Examples:

```
1234567890
-1234556790
+1234556790,432109876
```

5.6. FLOAT

"float": The "float" value type is used to express real numbers. If sign is not specified, the value is assumed positive "+". Multiple "float" values can be specified using the comma-separated notation.

Examples:

```
20.30
1000000.0000001
1.333,3.14
```

5.7. BINARY

"binary": The "binary" value type specifies that the type value is inline, encoded binary data. This value type can be specified in the PHOTO, LOGO, SOUND, and KEY types.

If inline encoded binary data is specified, the ENCODING type parameter MUST be used to specify the encoding format. The binary data MUST be encoded using the "B" encoding format. Long lines of encoded binary data SHOULD BE folded to 75 characters using the folding method defined in Section 4.1.

5.8. UTC-OFFSET

"utc-offset": The "utc-offset" value type specifies that the type value is a signed offset from UTC. This value type can be specified in the TZ type.

The value type is an offset from Coordinated Universal Time (UTC). It is specified as a positive or negative difference in units of hours and minutes (e.g., +hhmm). The time is specified as a 24-hour clock. Hour values are from 00 to 23, and minute values are from 00 to 59. Hour and minutes are 2-digits with high order zeroes required to maintain digit count. The basic format for ISO 8601 UTC offsets MUST be used.

5.9. LANGUAGE-TAG

"language-tag": A single language tag, as defined in [RFC4646].

6. Property Parameters

A property can have attributes associated with it. These "property parameters" contain meta-information about the property or the property value.

Property parameter values that contain the COLON (US-ASCII decimal 58), SEMICOLON (US-ASCII decimal 59) or COMMA (US-ASCII decimal 44) character separators MUST be specified as quoted-string text values. Property parameter values MUST NOT contain the DQUOTE (US-ASCII decimal 22) character. The DQUOTE (US-ASCII decimal 22) character is used as a delimiter for parameter values that contain restricted characters or URI text. For example:

```
EMAIL;PID=8:jdoe@example.com
```

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

Applications MUST ignore x-param and iana-param value they don't recognize.

6.1. LANGUAGE

The "language" property parameter is used to identify data in multiple languages. There is no concept of "default" language, except as specified by any "Content-Language" MIME header parameter that is present. The value of the "language" property parameter is a language tag as defined in Section 2 of [RFC4646].

ABNF:

```
language-param = "LANGUAGE=" Language-Tag
  ; Language-Tag is defined in section 2.1 of RFC 4646
```

6.2. ENCODING

The "encoding" property parameter is used to specify an alternate encoding for a value. If the value contains a CRLF, it must be encoded, since CRLF is used to separate lines in the content-type itself. Currently, only the "b" encoding is supported.

The "b" encoding can also be useful for binary values that are mixed with other text information in the body part (e.g., a certificate). Using a per-value "b" encoding in this case leaves the other information in a more readable form. The encoded base 64 value can be split across multiple physical lines by using the line folding technique described above.

The Content-Transfer-Encoding header field is used to specify the encoding used for the body part as a whole. The "encoding" property parameter is used to specify an encoding for a particular value (e.g., a certificate). In this case, the Content-Transfer-Encoding header might specify "8bit", while the one certificate value might specify an encoding of "b" via an "encoding=b" property parameter.

The Content-Transfer-Encoding and the encodings of individual properties given by the "encoding" property parameter are independent of one another. When encoding a text/vcard body part for transmission, individual property encodings are performed first, then the entire body part is encoded according to the Content-Transfer-Encoding. When decoding a text/vcard body part, the Content-Transfer-Encoding is decoded first, and then any individual properties with an "encoding" property parameter are decoded.

ABNF:

```
encoding-param = "ENCODING=" encoding-type

encoding-type = "b"          ; from [RFC2047]
              / iana-token   ; registered as in section 12
```

6.3. VALUE

The "value" parameter is optional, and is used to identify the value type (data type) and format of the value. The use of these predefined formats is encouraged even if the value parameter is not explicity used. By defining a standard set of value types and their formats, existing parsing and processing code can be leveraged. The predefined data type values MUST NOT be repeated in COMMA separated value lists except within the N, NICKNAME, ADR and CATEGORIES properties.

Including the value type explicitly as part of each property provides an extra hint to keep parsing simple and support more generalized applications. For example a search engine would not have to know the particular value types for all of the items for which it is searching. Because the value type is explicit in the definition, the search engine could look for dates in any item type and provide results that can still be interpreted.

ABNF:

```
value-param = "VALUE=" value-type

value-type = "text"
           / "uri"
           / "date"
           / "time"
           / "date-time"
           / "timestamp"
           / "boolean"
           / "integer"
           / "float"
           / "binary"
           / "utc-offset"
           / "language-tag"
           / iana-token  ; registered as described in section 12
           / x-name
```

6.4. PREF

The "pref" parameter is optional, and is used to indicate that the corresponding instance of a property is preferred by the vCard author. Its value MUST be an integer between 1 and 100 that quantifies the level of preferredness. Lower values correspond to a higher level of preferrredness, 1 being most preferred.

When the parameter is absent, the default MUST be to interpret the property instance as being least preferred.

Note that the value of this parameter is to be interpreted only in relation to values assigned to other instances of the same property in the same vCard. A given value, or the absence of a value, MUST NOT be interpreted on its own.

This parameter MAY be applied to any property that allows multiple instances.

ABNF:

```
pref-param = "PREF=" (1*2DIGIT / "100")
```

6.5. PID

The "pid" parameter is used to identify a specific property among multiple instances. It plays a role analogous to the UID property (Section 7.7.7) on a per-property instead of per-vCard basis. It MAY appear more than once in a given property. It MUST NOT appear on properties that only may have one instance per vCard. Its value is either a single small integer, or a pair of small integers separated by a dot. Multiple values may be encoded in a single PID parameter by separating the values with a comma ",". See Section 8 for more details on its usage.

ABNF:

```
pid-param = "PID=" pid-value *("," pid-value)
pid-value = 1*DIGIT ["." 1*DIGIT]
```

6.6. TYPE

The "type" parameter has multiple, different uses. In general, it is a way of specifying class characteristics of the associated property. Most of the time, its value is a comma-separated subset of a pre-defined enumeration. In this document, the following properties make use of this parameter: PHOTO, ADR, LABEL, TEL, EMAIL, IMPP, LOGO, MEMBER, SOUND, and KEY.

ABNF:

```
type-param= = "TYPE=" type-value *("," type-value)

type-value = type-value-tel / type-value-related
           / iana-token / x-name
  ; This is further defined in individual property sections.
```

7. vCard Properties

What follows is an enumeration of the standard vCard properties.

Property cardinalities are indicated using the following notation:

Cardinality	Meaning
(1,1)	Exactly one instance per vCard MUST be present.
(0,1)	Exactly one instance per vCard MAY be present.
(1,n)	One or more instances per vCard MUST be present.
(0,n)	One or more instances per vCard MAY be present.

7.1. General Properties

7.1.1. BEGIN

Purpose: To denote the beginning of a syntactic entity within a text/vcard content-type.

Value type: text

Cardinality: (1,1)

Special notes: The content entity MUST begin with the BEGIN property with a value of "VCARD".

The BEGIN property is used in conjunction with the END property to delimit an entity containing a related set of properties within an text/vcard content-type. This construct can be used instead of or in addition to wrapping separate sets of information inside additional MIME headers. It is provided for applications that wish to define content that can contain multiple entities within the same text/vcard content-type or to define content that can be identifiable outside of a MIME environment.

ABNF:

```
param =  ; no parameter allowed
value = "VCARD"
```

Example:

```
BEGIN:VCARD
```

7.1.2. END

Purpose: To denote the end of a syntactic entity within a text/vcard content-type.

Value type: text

Cardinality: (1,1)

Special notes: The content entity MUST end with the END type with a value of "VCARD".

The END property is used in conjunction with the BEGIN property to delimit an entity containing a related set of properties within an text/vcard content-type. This construct can be used instead of or in addition to wrapping separate sets of information inside additional MIME headers. It is provided for applications that wish to define content that can contain multiple entities within the same text/vcard content-type or to define content that can be identifiable outside of a MIME environment.

ABNF:

```
param =  ; no parameter allowed
value = "VCARD"
```

Example:

```
END:VCARD
```

7.1.3. SOURCE

Purpose: To identify the source of directory information contained in the content type.

Value type: uri

Cardinality: (0,n)

Special notes: The SOURCE property is used to provide the means by which applications knowledgable in the given directory service protocol can obtain additional or more up-to-date information from the directory service. It contains a URI as defined in [RFC3986] and/or other information referencing the vCard to which the information pertains. When directory information is available from more than one source, the sending entity can pick what it considers to be the best source, or multiple SOURCE properties can be included.

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Examples:

SOURCE:ldap://ldap.example.com/cn=Babs%20Jensen,%20o=Babsco,%20c=US

SOURCE:http://directory.example.com/addressbooks/jdoe/
Jean%20Dupont.vcf

7.1.4. NAME

Purpose: To identify the displayable name of the directory entity to which information in the vCard pertains.

Value type: text

Cardinality: (0,1)

Special notes: The NAME property is used to convey the display name of the entity to which the directory information pertains. Its value is the displayable, presentation text associated with the source for the vCard, as specified in the SOURCE property.

ABNF:

```
param = "VALUE=text" / any-param
value = text
```

Example:

```
NAME:Babs Jensen's Contact Information
```

7.1.5. KIND

Purpose: To specify the kind of object the vCard represents.

Value type: A single text value.

Cardinality: (0,1)

Special notes: The value may be one of: "individual" for a single person, "group" for a group of people, "org" for an organization, "location" for a named geographical place, an x-name or an iana-token. If this property is absent, "individual" MUST be assumed as default.

ABNF:

```
param = "VALUE=text" / any-param
value = "individual" / "group" / "org" / "location"
```

```
                / iana-token / x-name
```

Example:

This represents someone named Jane Doe working in the marketing department of the North American division of ABC Inc.

```
BEGIN:VCARD
VERSION:4.0
KIND:individual
FN:Jane Doe
ORG:ABC\, Inc.;North American Division;Marketing
END:VCARD
```

This represents the department itself, commonly known as ABC Marketing.

```
BEGIN:VCARD
VERSION:4.0
KIND:org
FN:ABC Marketing
ORG:ABC\, Inc.;North American Division;Marketing
END:VCARD
```

7.2. Identification Properties

These types are used to capture information associated with the identification and naming of the person or resource associated with the vCard.

7.2.1. FN

Purpose: To specify the formatted text corresponding to the name of the object the vCard represents.

Value type: A single text value.

Cardinality: (1,n)

Special notes: This property is based on the semantics of the X.520 Common Name attribute. The property MUST be present in the vCard object.

ABNF:

```
  param = "VALUE=text" / language-param / pid-param / pref-param
        / any-param
  value = text
```

Example:

FN:Mr. John Q. Public\, Esq.

7.2.2. N

Purpose: To specify the components of the name of the object the vCard represents.

Value type: A single structured text value. Each component can have multiple values.

Cardinality: (0,1)

Special note: The structured property value corresponds, in sequence, to the Surname, Given Names, Honorific Prefixes, and Honorific Suffixes. The text components are separated by the SEMI-COLON character (ASCII decimal 59). Individual text components can include multiple text values (e.g., multiple Additional Names) separated by the COMMA character (ASCII decimal 44). This property is based on the semantics of the X.520 individual name attributes. The property SHOULD be present in the vCard object when the name of the object the vCard represents follows the X.520 model.

ABNF:

```
param = "VALUE=text" / language-param / any-param
value = list-component 3(";" list-component)

list-component = list-component-value *("," list-component-value)
list-component-value = "\\" / "\," / "\;" / "\n" / WSP / NON-ASCII
                     / %x21-2B / %x2D-3A / %x3C-5B / %x5D-7E
```

Examples:

N:Public;John,Q.;Mr.;Esq.

N:Stevenson;John,Philip,Paul;Dr.;Jr.,M.D.,A.C.P.

7.2.3. NICKNAME

Purpose: To specify the text corresponding to the nickname of the object the vCard represents.

Value type: One or more text values separated by a COMMA character (ASCII decimal 44).

Cardinality: (0,n)

Special note: The nickname is the descriptive name given instead of or in addition to the one belonging to a person, place, or thing. It can also be used to specify a familiar form of a proper name specified by the FN or N properties.

ABNF:

```
  param = "VALUE=text" / language-param / pid-param / pref-param
        / any-param
  value = text-list
```

Examples:

```
NICKNAME:Robbie

NICKNAME:Jim,Jimmie

WORK.NICKNAME:Boss
```

7.2.4. PHOTO

Purpose: To specify an image or photograph information that annotates some aspect of the object the vCard represents.

Encoding: The encoding MUST be reset to "b" using the ENCODING parameter in order to specify inline, encoded binary data. If the value is referenced by a URI value, then the default encoding is used and no explicit ENCODING parameter is needed.

Value type: A single value. The default is binary value. It can also be reset to uri value. The uri value can be used to specify a value outside of this MIME entity.

Cardinality: (0,n)

Special notes: This property SHOULD include the parameter "TYPE" to specify the graphic image format type. The TYPE parameter value MUST be an image media type as specified in [RFC4288]. The full media type name, including the "image/" prefix, should be used. However, implementations SHOULD be able to handle bare subtypes.

ABNF:

```
  param = inline-param / refer-param
  value = inline-value / refer-value
    ; Value and parameter MUST match.

  param =/ pid-param / pref-param / any-param

  inline-param = "VALUE=binary"
               / encoding-param
               / "TYPE=" type-name "/" subtype-name
                   ; from [RFC4288] section 4.2
  inline-value = binary

  refer-param = "VALUE=uri"
  refer-value = uri
```

Example:

```
    PHOTO;VALUE=uri:http://www.example.com/pub/photos
     /jqpublic.gif


    PHOTO;ENCODING=b;TYPE=image/jpeg:MIICajCCAdOgAwIBAgICBEUwDQYJKo
     ZIhvcNAQEEBQAwdzELMAkGA1UEBhMCVVMxLDAqBgNVBAoTI05ldHNjYXBlIENv
     bW11bmljYXRpb25zIENvcnBvcmF0aW9uMRwwGgYDVQQLExNJbmZvcm1hdGlvbi
     <...remainder of "B" encoded binary data...>
```

7.2.5. BDAY

Purpose: To specify the birth date of the object the vCard represents.

Value type: The default is a single date value. It can also be reset to a single date-time or text value.

Cardinality: (0,1)

ABNF:

```
                param = "VALUE=" ("date" / "date-time" / "text")
                value = date / date-time / text
                ; Value and parameter MUST match.

                param =/ any-param
```

Examples:

```
BDAY:19960415
BDAY:--0415
BDAY;VALUE=date-time:19531015T231000Z
BDAY;VALUE=text:circa 1800
```

7.2.6. DDAY

Purpose: To specify the date of death of the object the vCard represents.

Value type: The default is a single date value. It can also be reset to a single date-time or text value.

Cardinality: (0,1)

ABNF:

```
param = "VALUE=" ("date" / "date-time" / "text")
value = date / date-time / text
; Value and parameter MUST match.

param =/ any-param
```

7.2.7. BIRTH

Purpose: To specify the place of birth of the object the vCard represents.

Value type: A single text value.

Cardinality: (0,1)

ABNF:

```
param = "VALUE=text" / language-param / any-param
value = text
```

Example:

```
BIRTH:Babies'R'Us Hospital
```

7.2.8. DEATH

Purpose: To specify the place of death of the object the vCard represents.

Value type: A single text value.

Cardinality: (0,1)

ABNF:

```
param = "VALUE=text" / language-param / any-param
value = text
```

Example:

```
DEATH:Aboard the Titanic\, near Newfoundland
```

7.2.9. GENDER

Purpose: To specify the gender of the object the vCard represents.

Value type: A single text value.

Cardinality: (0,1)

Special notes: The value "M" stands for male while "F" stands for female. IANA-registered values and x-names are also allowed.

ABNF:

```
param = "VALUE=text" / any-param
value = "M" / "F" / iana-token / x-name
```

Example:

```
GENDER:F
```

7.3. Delivery Addressing Properties

These types are concerned with information related to the delivery addressing or label for the vCard object.

7.3.1. ADR

Purpose: To specify the components of the delivery address for the vCard object.

Value type: A single structured text value, separated by the SEMI-COLON character (ASCII decimal 59).

Cardinality: (0,n)

Special notes: The structured type value consists of a sequence of address components. The component values MUST be specified in their corresponding position. The structured type value corresponds, in sequence, to the post office box; the extended address (e.g. apartment or suite number); the street address; the locality (e.g., city); the region (e.g., state or province); the postal code; the country name. When a component value is missing, the associated component separator MUST still be specified.

The text components are separated by the SEMI-COLON character (ASCII decimal 59). Where it makes semantic sense, individual text components can include multiple text values (e.g., a "street" component with multiple lines) separated by the COMMA character (ASCII decimal 44).

The property can include the "PREF" parameter to indicate the preferred delivery address when more than one address is specified.

The GEO parameter can be used to indicate global positioning information that is specific to this address. Its value is the same as that of the GEO property.

The TZ parameter can be used to indicate time zone information that is specific to this address. Its value is the same as that of the TZ property.

ABNF:

```
  param = "VALUE=text" / language-param / geo-param / tz-param
        / pid-param / pref-param / any-param
  value = list-component 6(";" list-component)

  geo-param = "GEO=" DQUOTE uri DQUOTE
  tz-param  = "TZ=" utc-offset
```

Example: In this example the post office box and the extended address are absent.

```
  ADR;GEO="geo:12.3457,78.910":;;123 Main Street;Any Town;CA
   ;91921-1234
```

7.3.2. LABEL

Purpose: To specify the formatted text corresponding to delivery address of the object the vCard represents.

Value type: A single text value.

Cardinality: (0,n)

Special notes: The property value is formatted text that can be used to present a delivery address label for the vCard object. The property can include the "PREF" parameter to indicate the preferred delivery address when more than one address is specified.

ABNF:

```
  param = "VALUE=text" / language-param / pid-param / pref-param
        / any-param
  value = text
```

Example: A multi-line address label.

```
        LABEL:Mr.John Q. Public\, Esq.\nMail Drop: TNE QB\n
         123 Main Street\nAny Town\, CA  91921-1234\nU.S.A.
```

7.4. Communications Properties

These properties are concerned with information associated with the way communications with the object the vCard represents are carried out.

7.4.1. TEL

Purpose: To specify the telephone number for telephony communication with the object the vCard represents.

Value type: A single URI value. It is expected that the URI scheme will be "tel", as specified in [RFC3966], but other schemes MAY be used.

Cardinality: (0,n)

Special notes: This property is based on the X.520 Telephone Number attribute.

The property can include the "PREF" parameter to indicate a prefered-use telephone number.

The property can include the parameter "TYPE" to specify intended use for the telephone number. The TYPE parameter values can include: "text" to indicate the telephone number supports text messages, "voice" to indicate a voice telephone number, "fax" to indicate a facsimile telephone number, "cell" to indicate a cellular telephone number, "video" to indicate a video conferencing telephone number, "pager" to indicate a paging device telephone number. The default type is "voice". These type parameter values can be specified as a parameter list (i.e., "TYPE=text;TYPE=voice") or as a value list (i.e., "TYPE=text,voice"). The default can be overridden to another set of values by specifying one or more alternate values. For example, the default TYPE of "voice" can be reset to a VOICE and FAX telephone number by the value list "TYPE=voice,fax".

ABNF:

```
param = "VALUE=uri" / type-param / pid-param / pref-param
      / any-param
value = uri

type-value-tel = "text" / "voice" / "fax" / "cell" / "video"
               / "pager" / iana-token / x-name
```

Example:

```
WORK.TEL;PREF=1;TYPE=voice,msg:tel:+1-555-555-5555;ext=5555
HOME.TEL:tel:+33-01-23-45-67
```

7.4.2. EMAIL

Purpose: To specify the electronic mail address for communication with the object the vCard represents.

Value type: A single text value.

Cardinality: (0,n)

Special notes: The property can include tye "PREF" parameter to indicate a preferred-use email address when more than one is specified.

ABNF:

```
param = "VALUE=text" / pid-param / pref-param / any-param
value = addr-spec  ; from [RFC5322] section 3.4.1
```

Type example:

```
WORK.EMAIL:jqpublic@xyz.example.com

EMAIL;PREF=1:jane_doe@example.com
```

7.4.3. IMPP

Purpose: To specify the URI for instant messaging and presence protocol communications with the object the vCard represents.

Value type: A single URI.

Cardinality: (0,n)

Special notes: The property may include the "PREF" parameter to indicate that this is a preferred address and has the same semantics as the "PREF" parameter in a TEL property.

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Example:

```
IMPP;PREF=1:xmpp:alice@example.com
```

7.4.4. LANG

Purpose: To specify the language(s) that may be used for contacting the individual associated with the vCard.

Value type: A single language-tag value.

Cardinality: (0,n)

ABNF:

```
param = "VALUE=language-tag" / pid-param / pref-param / any-param
value = Language-Tag
```

Example:

```
WORK.LANG;PREF=1:en
WORK.LANG;PREF=2:fr
HOME.LANG=fr
```

7.5. Geographical Properties

These properties are concerned with information associated with geographical positions or regions associated with the object the vCard represents.

7.5.1. TZ

Purpose: To specify information related to the time zone of the object the vCard represents.

Value type: The default is a single utc-offset value. It can also be reset to a single text value.

Cardinality: (0,n)

Special notes: The type value consists of a single value.

ABNF:

```
param = "VALUE=" ("utc-offset" / "text")
value = utc-offset / text
  ; Value and parameter must match

param =/ pid-param / pref-param / any-param
```

Type examples:

```
TZ:-0500

TZ;VALUE=text:-0500; EST; Raleigh/North America
;This example has a single value, not a structure text value.
```

7.5.2. GEO

Purpose: To specify information related to the global positioning of the object the vCard represents.

Value type: A single URI.

Cardinality: (0,n)

Special notes: The "geo" URI scheme [I-D.mayrhofer-geo-uri] is particularly well-suited for this property, but other schemes MAY be used.

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Example:

```
GEO:geo:37.386013,-122.082932
```

7.6. Organizational Properties

These properties are concerned with information associated with characteristics of the organization or organizational units of the object the vCard represents.

7.6.1. TITLE

Purpose: To specify the job title, functional position or function of the object the vCard represents.

Value type: A single text value.

Cardinality: (0,n)

Special notes: This property is based on the X.520 Title attribute.

ABNF:

```
param = "VALUE=text" / language-param / pid-param / pref-param
      / any-param
value = text
```

Example:

```
TITLE:Director\, Research and Development
```

7.6.2. ROLE

Purpose: To specify information concerning the role, occupation, or business category of the object the vCard represents.

Value type: A single text value.

Cardinality: (0,n)

Special notes: This property is based on the X.520 Business Category explanatory attribute. This property is included as an organizational type to avoid confusion with the semantics of the TITLE property and incorrect usage of that property when the semantics of this property is intended.

ABNF:

```
  param = "VALUE=text" / language-param / pid-param / pref-param
        / any-param
  value = text
```

Example:

```
        ROLE:Programmer
```

7.6.3. LOGO

Purpose: To specify a graphic image of a logo associated with the object the vCard represents.

Encoding: The encoding MUST be reset to "b" using the ENCODING parameter in order to specify inline, encoded binary data. If the value is referenced by a URI value, then the default encoding of 8bit is used and no explicit ENCODING parameter is needed.

Value type: A single value. The default is binary value. It can also be reset to uri value. The uri value can be used to specify a value outside of this MIME entity.

Cardinality: (0,n)

Special notes: This property SHOULD include the parameter "TYPE" to specify the graphic image format type. The TYPE parameter value MUST be an image media type as specified in [RFC4288]. The full media type name, including the "image/" prefix, should be used. However, implementations SHOULD be able to handle bare subtypes.

ABNF:

```
  param = inline-param / refer-param
  value = inline-value / refer-value
    ; Value and parameter MUST match.

  param =/ language-param / pid-param / pref-param / any-param
```

Example:

```
LOGO;VALUE=uri:http://www.example.com/pub/logos/abccorp.jpg

LOGO;ENCODING=b;TYPE=image/jpeg:MIICajCCAdOgAwIBAgICBEUwDQYJKoZ
 AQEEBQAwdzELMAkGA1UEBhMCVVMxLDAqBgNVBAoTI05ldHNjYXBlIENvbW11bm
 ljYXRpb25zIENvcnBvcmF0aW9uMRwwGgYDVQQLExNJbmZvcm1hdGlvbiBTeXN0
 <...the remainder of "B" encoded binary data...>
```

7.6.4. ORG

Purpose: To specify the organizational name and units associated with the vCard.

Value type: A single structured text value consisting of components separated the SEMI-COLON character (ASCII decimal 59).

Cardinality: (0,n)

Special notes: The property is based on the X.520 Organization Name and Organization Unit attributes. The property value is a structured type consisting of the organization name, followed by zero or more levels of organizational unit names.

ABNF:

```
  param = "VALUE=text" / language-param / pid-param / pref-param
        / any-param
  value = component *(";" component)

  component = "\\" / "\;" / "\n" / WSP / NON-ASCII
            / %x21-3A / %x3C-5B / %x5D-7E
```

Example: A property value consisting of an organizational name, organizational unit #1 name and organizational unit #2 name.

```
        ORG:ABC\, Inc.;North American Division;Marketing
```

7.6.5. MEMBER

Purpose: To include a member in the group this vCard represents.

Value tpe: A single URI. It MAY refer to something other than a vCard object. For example, an e-mail distribution list could employ the "mailto" URI scheme for efficiency.

Cardinality: (0,n)

Special notes: This property MUST NOT be present unless the value of the KIND property is "group".

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Examples:

```
BEGIN:VCARD
VERSION:4.0
KIND:group
FN:The Doe family
MEMBER:urn:uuid:03a0e51f-d1aa-4385-8a53-e29025acd8af
MEMBER:urn:uuid:b8767877-b4a1-4c70-9acc-505d3819e519
END:VCARD
BEGIN:VCARD
VERSION:4.0
FN:John Doe
UID:urn:uuid:03a0e51f-d1aa-4385-8a53-e29025acd8af
END:VCARD
BEGIN:VCARD
VERSION:4.0
FN:Jane Doe
UID:urn:uuid:b8767877-b4a1-4c70-9acc-505d3819e519
END:VCARD
```

```
BEGIN:VCARD
VERSION:4.0
KIND:group
FN:Funky distribution list
MEMBER:mailto:subscriber1@example.com
MEMBER:xmpp:subscriber2@example.com
MEMBER:sip:subscriber3@example.com
MEMBER:tel:+1-418-555-5555
END:VCARD
```

7.6.6. RELATED

Purpose: To specify a relationship the individual this vCard represents has with another.

Value type: A single URI. It can also be reset to a single text value. The text value can be used to specify textual information.

Cardinality: (0,n)

Special notes: The TYPE parameter MAY be used to characterize the related individual. The understood types are:

* "parent" means that the related individual is the parent of the individual this vCard represents.

* "child" means the opposite of "parent".

* "sibling" means that the two individuals are siblings.

* "manager" means that the related individual is the direct hierarchical superior (i.e. supervisor or manager) of the individual this vCard represents.

* "assistant" for an assistant or secretary.

* "agent" for a person who will act on behalf of the individual or resource associated with the vCard.

* "emergency" indicates an emergency contact.

Other types may be registered to IANA as described in Section 11.2, and private extensions starting with "X-" may be used.

ABNF:

```
  param = "VALUE=" ("uri" / "text")
  value = uri / text
    ; Parameter and value MUST match.

  param =/ type-param / pid-param / pref-param / any-param

  type-param-related = "parent" / "child" / "sibling" / "manager"
                     / "assistant" / "agent" / "emergency"
                     / iana-token / x-name
```

Examples:

```
RELATED;TYPE=manager:urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
RELATED;TYPE=assistant:http://example.com/directory/jdoe.vcf
RELATED;TYPE=agent;VALUE=text:Please contact my assistant Jane Doe
 for any inquiries.
```

7.7. Explanatory Properties

These properties are concerned with additional explanations, such as that related to informational notes or revisions specific to the vCard.

7.7.1. CATEGORIES

Purpose: To specify application category information about the vCard.

Value type: One or more text values separated by a COMMA character (ASCII decimal 44).

Cardinality: (0,n)

ABNF:

```
param = "VALUE=text" / pid-param / pref-param / any-param
value = text-list
```

Example:

```
CATEGORIES:TRAVEL AGENT

CATEGORIES:INTERNET,IETF,INDUSTRY,INFORMATION TECHNOLOGY
```

7.7.2. NOTE

Purpose: To specify supplemental information or a comment that is associated with the vCard.

Value type: A single text value.

Cardinality: (0,n)

Special notes: The property is based on the X.520 Description attribute.

ABNF:

```
param = "VALUE=text" / language-param / pid-param / pref-param
      / any-param
```

```
  value = text
```

Example:

```
        NOTE:This fax number is operational 0800 to 1715
          EST\, Mon-Fri.
```

7.7.3. PRODID

Purpose: To specify the identifier for the product that created the vCard object.

Type value: A single text value.

Cardinality: (0,1)

Special notes: Implementations SHOULD use a method such as that specified for Formal Public Identifiers in [ISO9070] or for Universal Resource Names in [RFC3406] to assure that the text value is unique.

ABNF:

```
  param = "VALUE=text" / any-param
  value = text
```

Example:

```
        PRODID:-//ONLINE DIRECTORY//NONSGML Version 1//EN
```

7.7.4. REV

Purpose: To specify revision information about the current vCard.

Value type: A single timestamp value.

Cardinality: (0,1)

Special notes: The value distinguishes the current revision of the information in this vCard for other renditions of the information.

ABNF:

```
  param = "VALUE=timestamp" / any-param
  value = timestamp
```

Example:

```
REV:19951031T222710Z
```

7.7.5. SORT-STRING

Purpose: To specify the family name or given name text to be used for national-language-specific sorting of the FN and N types.

Value type: A single text value.

Cardinality: (0,1)

Special notes: The sort string is used to provide family name or given name text that is to be used in locale- or national-language- specific sorting of the formatted name and structured name types. Without this information, sorting algorithms could incorrectly sort this vCard within a sequence of sorted vCards. When this property is present in a vCard, then this family name or given name value is used for sorting the vCard.

ABNF:

```
param = "VALUE=text" / any-param
value = text
```

Examples: For the case of family name sorting, the following examples define common sort string usage with the FN and N properties.

```
FN:Rene van der Harten
N:van der Harten;Rene;J.;Sir;R.D.O.N.
SORT-STRING:Harten

FN:Robert Pau Shou Chang
N:Pau;Shou Chang;Robert
SORT-STRING:Pau

FN:Osamu Koura
N:Koura;Osamu
SORT-STRING:Koura

FN:Oscar del Pozo
N:del Pozo Triscon;Oscar
SORT-STRING:Pozo

FN:Chistine d'Aboville
N:d'Aboville;Christine
SORT-STRING:Aboville
```

7.7.6. SOUND

Purpose: To specify a digital sound content information that annotates some aspect of the vCard. This property is often used to specify the proper pronunciation of the name property value of the vCard.

Encoding: The encoding MUST be reset to "b" using the ENCODING parameter in order to specify inline, encoded binary data. If the value is referenced by a URI value, then the default encoding of 8bit is used and no explicit ENCODING parameter is needed.

Value type: A single value. The default is binary value. It can also be reset to uri value. The uri value can be used to specify a value outside of this MIME entity.

Cardinality: (0,n)

Special notes: This property SHOULD include the parameter "TYPE" to specify the audio format type. The TYPE parameter value MUST be an audio media type as specified in [RFC4288]. The full media type name, including the "audio/" prefix, should be used. However, implementations SHOULD be able to handle bare subtypes.

ABNF:

```
  param = inline-param / refer-param
  value = inline-value / refer-value
    ; Value and parameter MUST match.

  param =/ language-param / pid-param / pref-param / any-param
```

Example:

```
SOUND;TYPE=audio/basic;VALUE=uri:CID:JOHNQPUBLIC.part8.
 19960229T080000.xyzMail@example.com

SOUND;TYPE=audio/basic;ENCODING=b:MIICajCCAdOgAwIBAgICBEUwDQYJK
 AQEEBQAwdzELMAkGA1UEBhMCVVMxLDAqBgNVBAoTI05ldHNjYXBlIENvbW11bm
 ljYXRpb25zIENvcnBvcmF0aW9uMRwwGgYDVQQLExNJbmZvcm1hdGlvbiBTeXN0
 <...the remainder of "B" encoded binary data...>
```

7.7.7. UID

Purpose: To specify a value that represents a globally unique identifier corresponding to the individual or resource associated with the vCard.

Value type: A single URI value.

Cardinality: (0,1)

Special notes: This property is used to uniquely identify the object that the vCard represents. The "uuid" URN namespace defined in [RFC4122] is particularly well-suited to this task, but other URI schemes MAY be used.

ABNF:

```
param = "VALUE=uri" / any-param
value = uri
```

Example:

```
UID:urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

7.7.8. CLIENTPIDMAP

Purpose: To give a global meaning to a local PID source identifier.

Value type: A semicolon-separated pair of values. The first field is a small integer corresponding to the second field of a PID parameter instance. The second field is a URI. The "uuid" URN namespace defined in [RFC4122] is particularly well-suited to this task, but other URI schemes MAY be used.

Cardinality: (0,n)

Special notes: PID source identifiers (the source identifier is the second field in a PID parameter instance) are small integers that only have significance within the scope of a single vCard instance. Each distinct source identifier present in a vCard MUST have an associated CLIENTPIDMAP. See Section 8 for more details on the usage of CLIENTPIDMAP.

PID source identifiers MUST be strictly positive. Zero is not allowed.

As a special exception, the PID parameter MUST NOT be applied to this property.

ABNF:

```
param = any-param
value = 1*DIGIT ";" uri
```

Example:

```
TEL;PID=3.1,4.2:tel:+1-555-555-5555
EMAIL;PID=4.1,5.2:jdoe@example.com
CLIENTPIDMAP:1;urn:uuid:3df403f4-5924-4bb7-b077-3c711d9eb34b
CLIENTPIDMAP:2;urn:uuid:d89c9c7a-2e1b-4832-82de-7e992d95faa5
```

7.7.9. URL

Purpose: To specify a uniform resource locator associated with the object that the vCard refers to.

Cardinality: (0,n)

Value type: A single uri value.

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Example:

```
URL:http://example.org/restaurant.french/~chezchic.html
```

7.7.10. VERSION

Purpose: To specify the version of the vCard specification used to format this vCard.

Value type: A single text value.

Cardinality: (1,1)

Special notes: The property MUST be present in the vCard object. The value MUST be "4.0" if the vCard corresponds to this specification.

ABNF:

```
param = "VALUE=text" / any-param
value = "4.0"
```

Example:

```
VERSION:4.0
```

7.8. Security Properties

These properties are concerned with the security of communication pathways or access to the vCard.

7.8.1. CLASS

Purpose: To specify the access classification for a vCard object.

Value type: A single text value.

Cardinality: (0,1)

Special notes: An access classification is only one component of the general security model for a directory service. The classification attribute provides a method of capturing the intent of the owner for general access to information described by the vCard object.

Predefined values are:

PUBLIC: This vCard MAY be shared with anyone.

PRIVATE: This vCard MUST NOT be shared. It MAY be exported if explictly authorized and requested by the creator.

CONFIDENTIAL: This vCard MAY be shared with allowed users or systems. The exact confidentiality level is site-specific and out of scope for the vCard specification.

ABNF:

```
param = "VALUE=text" / any-param
value = "PUBLIC" / "PRIVATE" / "CONFIDENTIAL" / iana-token / x-name
```

Examples:

```
CLASS:PUBLIC

CLASS:PRIVATE

CLASS:CONFIDENTIAL
```

7.8.2. KEY

Purpose: To specify a public key or authentication certificate associated with the object that the vCard represents.

Encoding: The encoding MUST be reset to "b" using the ENCODING parameter in order to specify inline, encoded binary data. If the value is a text value, then the default encoding of 8bit is used and no explicit ENCODING parameter is needed.

Value type: A single value. The default is binary. It can also be reset to uri value. The uri value can be used to specify a value outside of this MIME entity. In this case, the key's media type is obtained externally (e.g. with the HTTP Content-Type header) instead of with the TYPE parameter.

Cardinality: (0,n)

Special notes: This property SHOULD include the parameter "TYPE" to specify the public key or authentication certificate format. The TYPE parameter value MUST be a media type as specified in [RFC4288].

ABNF:

```
param = inline-param / refer-param
value = inline-value / refer-value
  ; Value and parameter MUST match.

param =/ pid-param / pref-param / any-param
```

Examples:

```
KEY;VALUE=uri:http://www.example.com/keys/jdoe
```

```
KEY;TYPE=application/pgp-keys;ENCODING=b:mQGiBEbEPUsRBACBF0RSIN
 mGutdM+KSAl7HMzwXHaLbvEOyu8At80I8qGejhzWowKbfem3X0m68Y/vhb+J2g
 7q11KHpnEdNb67uZaj9nTQ09Q+UFtH25qD/Afn3+9bOJQaPjAUYzXu3vD/xmN8
 <...remainder of "B" encoded binary data...>
```

7.9. Calendar Properties

These properties are further specified in [RFC2739].

7.9.1. FBURL

Purpose: To specify the URI for a user's busy time in a vCard object.

Value type: A single URI value.

Cardinality: (0,n)

Special notes: Where multiple FBURL properties are specified, the default FBURL property is indicated with the PREF parameter. The FTP or HTTP type of URI points to an iCalendar object associated with a snapshot of the last six weeks of the user's busy time data. If the iCalendar object is represented as a file or document, it's file type should be "ifb".

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Examples:

```
FBURL;PREF=1:http://www.example.com/busy/janedoe
FBURL:FTP://ftp.example.com/busy/project-a.ifb
```

7.9.2. CALADRURI

Purpose: To specify the location to which an event request should be sent for the user.

Value type: A single URI value.

Cardinality: (0,n)

Special notes: Where multiple CALADRURI properties are specified, the default CALADRURI property is indicated with the PREF parameter.

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Example:

```
CALADRURI;PREF=1:mailto:janedoe@example.com
CALADRURI:http://example.com/calendar/jdoe
```

7.9.3. CALURI

Purpose: To specify the URI for a user's calendar in a vCard object.

Value type: A single URI value.

Cardinality: (0,n)

Special notes: Where multiple CALURI properties are specified, the default CALURI property is indicated with the PREF parameter. The property should contain a URI pointing to an iCalendar object associated with a snapshot of the user's calendar store. If the iCalendar object is represented as a file or document, it's file type should be "ics".

ABNF:

```
param = "VALUE=uri" / pid-param / pref-param / any-param
value = uri
```

Examples:

```
CALURI;PREF=1:http://cal.example.com/calA
CALURI:ftp://ftp.example.com/calA.ics
```

7.10. Extended Properties and Parameters

The properties and parameters defined by this document can be extended. Non-standard, private properties and parameters with a name starting with "X-" may be defined bilaterally between two cooperating agents without outside registration or standardization.

8. Synchronization

vCard data often needs to be synchronized between devices. In this context, synchronization is defined as the intelligent merging of two representations of the same object. vCard 4.0 includes mechanisms to aid this process.

8.1. Mechanisms

Two mechanisms are available: the UID property is used to match multiple instances of the same vCard, while the PID parameter is used to match multiple instances of the same property.

The term "matching" is used here to mean recognizing that two instances are in fact representations of the same object. For example, a single vCard that is shared with someone results in two

vCard instances. After they have evolved separately, they still represent the same object, and therefore may be matched by a synchronization engine.

8.1.1. Matching vCard Instances

vCard instances for which the UID properties (Section 7.7.7) are equivalent MUST be matched. Equivalence is determined as specified in [RFC3986], Section 6.

In all other cases, vCard instances MAY be matched at the discretion of the synchronization engine.

8.1.2. Matching Property Instances

Property instances belonging to unmatched vCards MUST NOT be matched.

Property instances whose name (e.g. EMAIL, TEL, etc.) is not the same MUST NOT be matched.

Property instances whose name is CLIENTPIDMAP are handled separately and MUST NOT be matched. The synchronization MUST ensure that there is consistency of CLIENTPIDMAPs among matched vCard instances.

Property instances belonging to matched vCards, whose name is the same, and whose maximum cardinality is 1 MUST be matched.

Property instances belonging to matched vCards, whose name is the same, and whose PID parameters match MUST be matched. See Section 8.1.3 for details on PID matching.

In all other cases, property instances MAY be matched at the discretion of the synchronization engine.

8.1.3. PID Matching

Two PID values for which the first fields are equivalent represent the same local value.

Two PID values representing the same local value and for which the second fields point to CLIENTPIDMAP properties whose second field URIs are equivalent (as specified in [RFC3986], Section 6) also represent the same global value.

PID parameters for which at least one pair of their values represent the same global value MUST be matched.

In all other cases, PID parameters MAY be matched at the discretion

of the synchronization engine.

For example, PID value "5.1", in the first vCard below, and PID value "6.2", in the second vCard below, represent the same global value.

```
BEGIN:VCARD
VERSION:4.0
EMAIL;PID=4.2,5.1:jdoe@example.com
CLIENTPIDMAP:1;urn:uuid:3eef374e-7179-4196-a914-27358c3e6527
CLIENTPIDMAP:2;urn:uuid:42bcd5a7-1699-4514-87b4-056edf68e9cc
END:VCARD

BEGIN:VCARD
VERSION:4.0
EMAIL;PID=5.1,6.2:john@example.com
CLIENTPIDMAP:1;urn:uuid:0c75c629-6a8d-4d5e-a07f-1bb35846854d
CLIENTPIDMAP:2;urn:uuid:3eef374e-7179-4196-a914-27358c3e6527
END:VCARD
```

8.2. Example

8.2.1. Creation

The following simple vCard is first created on a given device.

```
BEGIN:VCARD
VERSION:4.0
UID:urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1
FN:J. Doe
EMAIL;PID=1.1:jdoe@example.com
CLIENTPIDMAP:1;urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556
END:VCARD
```

This new vCard is assigned the UID "urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1" by the creating device. The EMAIL property is assigned PID 1, and this PID is given global context by associating it with "urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556", which represents the creating device. The FN property has no PID because it is forbidden by its maximum cardinality of 1.

8.2.2. Initial Sharing

This vCard is shared with a second device. Upon inspecting the UID property, the second device understands that this is a new vCard (i.e. unmatched) and thus the synchronization results in a simple copy.

8.2.3. Adding and Sharing a Property

A new phone number is created on the first device, then the vCard is shared with the second device. This is what the second device receives:

```
BEGIN:VCARD
VERSION:4.0
UID:urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1
FN:J. Doe
EMAIL;PID=1.1:jdoe@example.com
TEL;PID=1.1:tel:+1-555-555-5555
CLIENTPIDMAP:1;urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556
END:VCARD
```

Upon inspecting the UID property, the second device matches the vCard it received to the vCard that it already has stored. It then starts comparing the properties of the two vCards in same-named pairs.

The FN properties are matched automatically because their maximum cardinality is 1. Since the property value is the same, no update takes place.

The EMAIL properties are matched because the PID parameters have the same global value. Since the property value is the same, no update takes place.

The TEL property in the new vCard is not matched to any in the stored vCard because no property in the stored vCard has the same name. Therefore, this property is copied from the new vCard to the stored vCard.

The CLIENTPIDMAP property is handled separately by the synchronization engine. It ensures that it is consistent with the stored one. If it was not, the results would be up to the synchronization engine, and thus undefined by this document.

8.2.4. Simultaneous Editing

A new email address and a new phone number are added to the vCard on each of the two devices, and then a new synchronization event happens. Here are the vCards that are communicated to each other:

```
BEGIN:VCARD
VERSION:4.0
UID:urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1
FN:J. Doe
EMAIL;PID=1.1:jdoe@example.com
EMAIL;PID=2.1:boss@example.com
TEL;PID=1.1:tel:+1-555-555-5555
TEL;PID=2.1:tel:+1-666-666-6666
CLIENTPIDMAP:1;urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556
END:VCARD

BEGIN:VCARD
VERSION:4.0
UID:urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1
FN:J. Doe
EMAIL;PID=1.1:jdoe@example.com
EMAIL;PID=2.2:ceo@example.com
TEL;PID=1.1:tel:+1-555-555-5555
TEL;PID=2.2:tel:+1-666-666-6666
CLIENTPIDMAP:1;urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556
CLIENTPIDMAP:2;urn:uuid:1f762d2b-03c4-4a83-9a03-75ff658a6eee
END:VCARD
```

On the first device, the same PID source identifier (1) is reused for the new EMAIL and TEL properties. On the second device, a new source identifier (2) is generated, and a corresponding CLIENTPIDMAP property is created. It contains the second device's identifier, "urn:uuid:1f762d2b-03c4-4a83-9a03-75ff658a6eee".

The new EMAIL properties are unmatched on both sides since the PID global value is new in both cases. The sync thus results in a copy on both sides.

Although the situation appears to be the same for the TEL properties, in this case the synchronization engine is particularly smart and matches the two new TEL properties even though their PID global values are different. Note that in this case, the rules of section Section 8.1.2 state that two properties may be matched at the discretion of the synchronization engine. Therefore, the two properties are merged.

All this results in the following vCard which is stored on both devices:

```
BEGIN:VCARD
VERSION:4.0
UID:urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1
FN:J. Doe
EMAIL;PID=1.1:jdoe@example.com
EMAIL;PID=2.1:boss@example.com
EMAIL;PID=2.2:ceo@example.com
TEL;PID=1.1:tel:+1-555-555-5555
TEL;PID=2.1,2.2:tel:+1-666-666-6666
CLIENTPIDMAP:1;urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556
CLIENTPIDMAP:2;urn:uuid:1f762d2b-03c4-4a83-9a03-75ff658a6eee
END:VCARD
```

8.2.5. Global Context Simplification

The two devices finish their synchronization procedure by simplifying their global contexts. Since they haven't talked to any other device, the following vCard is for all purposes equivalent to the above. It is also shorter.

```
BEGIN:VCARD
VERSION:4.0
UID:urn:uuid:4fbe8971-0bc3-424c-9c26-36c3e1eff6b1
FN:J. Doe
EMAIL;PID=1.1:jdoe@example.com
EMAIL;PID=2.1:boss@example.com
EMAIL;PID=3.1:ceo@example.com
TEL;PID=1.1:tel:+1-555-555-5555
TEL;PID=2.1:tel:+1-666-666-6666
CLIENTPIDMAP:1;urn:uuid:53e374d9-337e-4727-8803-a1e9c14e0556
END:VCARD
```

The details of global context simplification are unspecified by this document. They are left up to the synchronization engine. This example is merely intended to illustrate the possibility, which investigating would be, in the authors' opinion, worthwhile.

9. Example: Authors' vCards

```
BEGIN:VCARD
VERSION:4.0
FN:Simon Perreault
N:Perreault;Simon;;ing. jr,M.Sc.
BDAY:--0203
GENDER:M
LANG;PREF=1:fr
LANG;PREF=2:en
WORK.ORG:Viagenie
WORK.ADR:;Suite 625;2600 boul. Laurier;
 Quebec;QC;G1V 4W1;Canada
WORK.TEL;TYPE=voice;PREF=1:tel:+1-418-656-9254;ext=102
WORK.TEL;TYPE=cell,voice,video,text:tel:+1-418-262-6501
WORK.TEL;TYPE=fax:tel:+1-418-656-9257
WORK.EMAIL:simon.perreault@viagenie.ca
WORK.GEO:geo:46.772673,-71.282945
WORK.KEY;VALUE=uri:http://www.viagenie.ca/simon.perreault/simon.asc
TZ:-0500
CLASS:PUBLIC
END:VCARD


BEGIN:VCARD
VERSION:4.0
FN:Pete Resnick
N:Resnick;Pete;;
GENDER:M
WORK.ORG:QUALCOMM Incorporated
WORK.ADR:;;5775 Morehouse Drive;San Diego;CA;92121-1714;US
WORK.TEL;TYPE=voice:tel:+1-858-651-4478
WORK.EMAIL:presnick@qualcomm.com
WORK.URL:http://www.qualcomm.com/~presnick/
END:VCARD
```

10. Security Considerations

- Internet mail is subject to many well known security attacks, including monitoring, replay, and forgery. Care should be taken by any directory service in allowing information to leave the scope of the service itself, where any access controls can no longer be guaranteed. Applications should also take care to display directory data in a "safe" environment (e.g., PostScript-valued types).

- vCards can carry cryptographic keys or certificates, as described in Section 7.8.2.

- Section 7.8.1 specifies a desired security classification policy for a particular vCard. That policy is not enforced in any way.

- The vCard objects have no inherent authentication or privacy, but can easily be carried by any security mechanism that transfers MIME objects with authentication or privacy. In cases where threats of "spoofed" vCard information is a concern, the vCard SHOULD BE transported using one of these secure mechanisms.

- The information in a vCard may become out of date. In cases where the vitality of data is important to an originator of a vCard, the "URL" type described in Section 7.7.9 SHOULD BE specified. In addition, the "REV" type described in section Section 7.7.4 can be specified to indicate the last time that the vCard data was updated.

11. IANA Considerations

11.1. MIME Type Registration

Please refer to Section 3.

11.2. Registering New vCard Elements

This section defines the process for registering new or modified vCard elements (i.e. properties, parameters, value data types, and values) with IANA.

11.2.1. Registration Procedure

The IETF will create a mailing list, vcard@ietf.org [1], which can be used for public discussion of vCard element proposals prior to registration. Use of the mailing list is strongly encouraged. The IESG will appoint a designated expert who will monitor the vcard@ietf.org [1] mailing list and review registrations.

Registration of new vCard elements MUST be reviewed by the designated expert and published in an RFC. A Standard Tracks RFC is REQUIRED for the regisration of new value data types that modify existing properties. A Standard Tracks RFC is also REQUIRED for registration of vCard elements that modify vCard elements previously documented in a Standard Tracks RFC.

The registration procedure begins when a completed registration template, defined in the sections below, is sent to vcard@ietf.org [1] and iana@iana.org [2]. 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.

11.2.2. Vendor Namespace

The vendor namespace is used for vCard elements associated with commercially available products. "Vendor" or "producer" are construed as equivalent and very broadly in this context.

A registration may be placed in the vendor namespace by anyone who needs to interchange files associated with the particular product. However, the registration formally belongs to the vendor or organization handling the vCard elements in the namespace being registered. Changes to the specification will be made at their request, as discussed in subsequent sections.

vCard elements belonging to the vendor namespace will be distinguished by the "VND-" prefix. That may be followed, at the discretion of the registrant, by either a vCard element name from a well-known producer (e.g., "VND-MUDPIE") or by an IANA-approved designation of the producer's name that is followed by a vCard element designation (e.g., "VND-BIGCOMPANY-MUDPIE").

While public exposure and review of vCard elements to be registered in the vendor namespace is not required, using the vcard@ietf.org [1] mailing list for review is strongly encouraged to improve the quality of those specifications. Registrations in the vendor namespace may be submitted directly to the IANA.

11.2.3. Registration Template for Groups

A group is defined by completing the following template.

Group name: The name of the group.

Purpose: The purpose of the group. Give a short but clear description.

Description: Any special notes about the group, how it is to be used, etc.

Allowed Properties: List of properties that may be placed inside the group.

Example(s): One or more examples of instances of the value type needs to be specified.

11.2.4. Registration Template for Properties

A property is defined by completing the following template.

Property name: The name of the property.

Purpose: The purpose of the property. Give a short but clear description.

Value type: Any of the valid value types for the property value needs to be specified. The default value type also needs to be specified.

Property parameters: Any of the valid property parameters for the property MUST be specified.

Groups: List of already-existing groups whose allowed properties list must be updated by adding this new property.

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

Format definition: The ABNF for the property definition needs to be specified.

Example(s): One or more examples of instances of the property needs to be specified.

11.2.5. Registration Template for Parameters

A parameter is defined by completing the following template.

Parameter name: The name of the parameter.

Purpose: The purpose of the parameter. Give a short but clear description.

Description: Any special notes about the parameter, how it is to be used, etc.

Format definition: The ABNF for the parameter definition needs to be specified.

Example(s): One or more examples of instances of the parameter needs to be specified.

11.2.6. Registration Template for Value Data Types

A value data type is defined by completing the following template.

Value name: The name of the value type.

Purpose: The purpose of the value type. Give a short but clear description.

Description: Any special notes about the value type, how it is to be used, etc.

Format definition: The ABNF for the value type definition needs to be specified.

Example(s): One or more examples of instances of the value type needs to be specified.

11.2.7. Registration Template for Values

A value is defined by completing the following template.

Value: The value literal.

Purpose: The purpose of the value. Give a short but clear description.

Conformance: The vCard properties and/or parameters that can take this value needs to be specified.

Example(s): One or more examples of instances of the value needs to be specified.

The following is a fictitious example of a registration of a vCard value:

Value: TOP-SECRET

Purpose: This value is used to specify the access classification of top-secret vCards.

Conformance: This value can be used with the "CLASS" property.

Example(s): The following is an example of this value used with the "CLASS" property:

```
CLASS:TOP-SECRET
```

11.3. Initial vCard Elements Registries

The IANA is requested to create and maintain the following registries for vCard elements with pointers to appropriate reference documents.

11.3.1. Groups Registry

The following table is to be used to initialize the groups registry.

Goup	Description	Status	Allowed Properties	Reference
WORK	Properties related to an individual's work place.	Current	FN, NICKNAME, PHOTO, ADR, LABEL, TEL, EMAIL, IMPP, LANG, TZ, GEO, TITLE, ROLE, LOGO, ORG, RELATED, CATEGORIES, NOTE, SOUND, URL, KEY, FBURL, CALADRURI, CALURI	RFCXXXX
HOME	Properties related to an individual's personal life.	Current	FN, NICKNAME, PHOTO, ADR, LABEL, TEL, EMAIL, IMPP, LANG, TZ, GEO, TITLE, ROLE, LOGO, ORG, RELATED, CATEGORIES, NOTE, SOUND, URL, KEY, FBURL, CALADRURI, CALURI	RFCXXXX

11.3.2. Properties Registry

The following table is to be used to initialize the properties registry.

Property	Status	Reference
SOURCE	Current	RFCXXXX, Section 7.1.3
NAME	Current	RFCXXXX, Section 7.1.4
KIND	Current	RFCXXXX, Section 7.1.5
FN	Current	RFCXXXX, Section 7.2.1
N	Current	RFCXXXX, Section 7.2.2
NICKNAME	Current	RFCXXXX, Section 7.2.3
PHOTO	Current	RFCXXXX, Section 7.2.4
BDAY	Current	RFCXXXX, Section 7.2.5
DDAY	Current	RFCXXXX, Section 7.2.6
BIRTH	Current	RFCXXXX, Section 7.2.7
DEATH	Current	RFCXXXX, Section 7.2.8
GENDER	Current	RFCXXXX, Section 7.2.9
ADR	Current	RFCXXXX, Section 7.3.1
LABEL	Current	RFCXXXX, Section 7.3.2
TEL	Current	RFCXXXX, Section 7.4.1
EMAIL	Current	RFCXXXX, Section 7.4.2
IMPP	Current	RFCXXXX, Section 7.4.3
LANG	Current	RFCXXXX, Section 7.4.4
TZ	Current	RFCXXXX, Section 7.5.1
GEO	Current	RFCXXXX, Section 7.5.2
TITLE	Current	RFCXXXX, Section 7.6.1
ROLE	Current	RFCXXXX, Section 7.6.2
LOGO	Current	RFCXXXX, Section 7.6.3
ORG	Current	RFCXXXX, Section 7.6.4
MEMBER	Current	RFCXXXX, Section 7.6.5
RELATED	Current	RFCXXXX, Section 7.6.6
CATEGORIES	Current	RFCXXXX, Section 7.7.1
NOTE	Current	RFCXXXX, Section 7.7.2
PRODID	Current	RFCXXXX, Section 7.7.3
REV	Current	RFCXXXX, Section 7.7.4
SORT-STRING	Current	RFCXXXX, Section 7.7.5
SOUND	Current	RFCXXXX, Section 7.7.6
UID	Current	RFCXXXX, Section 7.7.7
CLIENTPIDMAP	Current	RFCXXXX, Section 7.7.8
URL	Current	RFCXXXX, Section 7.7.9
VERSION	Current	RFCXXXX, Section 7.7.10
CLASS	Current	RFCXXXX, Section 7.8.1
KEY	Current	RFCXXXX, Section 7.8.2
FBURL	Current	RFCXXXX, Section 7.9.1
CALADRURI	Current	RFCXXXX, Section 7.9.2
CALURI	Current	RFCXXXX, Section 7.9.3

11.3.3. Parameters Registry

The following table is to be used to initialize the parameters registry.

Parameter	Status	Reference
LANGUAGE	Current	RFCXXXX, Section 6.1
ENCODING	Current	RFCXXXX, Section 6.2
VALUE	Current	RFCXXXX, Section 6.3
PREF	Current	RFCXXXX, Section 6.4
PID	Current	RFCXXXX, Section 6.5
TYPE	Current	RFCXXXX, Section 6.6
GEO	Current	RFCXXXX, Section 7.3.1
TZ	Current	RFCXXXX, Section 7.3.1

11.3.4. Value Data Types Registry

The following table is to be used to initialize the parameters registry.

Value Data Type	Status	Reference
BINARY	Current	RFCXXXX, Section 5.7
BOOLEAN	Current	RFCXXXX, Section 5.4
DATE	Current	RFCXXXX, Section 5.3.1
TIME	Current	RFCXXXX, Section 5.3.2
DATE-TIME	Current	RFCXXXX, Section 5.3.3
TIMESTAMP	Current	RFCXXXX, Section 5.3.4
FLOAT	Current	RFCXXXX, Section 5.6
INTEGER	Current	RFCXXXX, Section 5.5
TEXT	Current	RFCXXXX, Section 5.1
URI	Current	RFCXXXX, Section 5.2
LANGUAGE-TAG	Current	RFCXXXX, Section 5.9

11.3.5. Values Registries

Separate tables will be used for property and parameter values.

The following table is to be used to initialize the property values registry.

Property	Value	Status	Reference
BEGIN	VCARD	Current	RFCXXXX, Section 7.1.1
END	VCARD	Current	RFCXXXX, Section 7.1.2
KIND	individual	Current	RFCXXXX, Section 7.1.5
KIND	group	Current	RFCXXXX, Section 7.1.5
KIND	org	Current	RFCXXXX, Section 7.1.5
KIND	location	Current	RFCXXXX, Section 7.1.5
GENDER	M	Current	RFCXXXX, Section 7.2.9
GENDER	F	Current	RFCXXXX, Section 7.2.9
CLASS	PUBLIC	Current	RFCXXXX, Section 7.8.1
CLASS	PRIVATE	Current	RFCXXXX, Section 7.8.1
CLASS	CONFIDENTIAL	Current	RFCXXXX, Section 7.8.1

The following table is to be used to initialize the parameter values registry.

Property	Parameter	Value	Status	Reference
TEL	TYPE	text	Current	RFCXXXX, Section 7.4.1
TEL	TYPE	voice	Current	RFCXXXX, Section 7.4.1
TEL	TYPE	fax	Current	RFCXXXX, Section 7.4.1
TEL	TYPE	cell	Current	RFCXXXX, Section 7.4.1
TEL	TYPE	video	Current	RFCXXXX, Section 7.4.1
TEL	TYPE	pager	Current	RFCXXXX, Section 7.4.1
RELATED	TYPE	parent	Current	RFCXXXX, Section 7.6.6
RELATED	TYPE	child	Current	RFCXXXX, Section 7.6.6
RELATED	TYPE	sibling	Current	RFCXXXX, Section 7.6.6
RELATED	TYPE	manager	Current	RFCXXXX, Section 7.6.6
RELATED	TYPE	assistant	Current	RFCXXXX, Section 7.6.6
RELATED	TYPE	agent	Current	RFCXXXX, Section 7.6.6

12. Acknowledgements

The authors would like to thank Tim Howes, Mark Smith, and Frank Dawson, the original authors of [RFC2425] and [RFC2426], as well as the following individuals who have participated in the drafting, review and discussion of this memo:

Marc Blanchet, Stephane Bortzmeyer, Dan Brickley, Chris Bryant, Dany Cauchie, Darryl Champagne, Cyrus Daboo, Bernard Desruisseaux, Lisa Dusseault, Javier Godoy, Helge Hess, Alexander Mayrhofer, Chris Newman, Mark Paterson, Julian Reschke, Peter K. Sheerin, Anil Srivastava, and Kurt Zeilenga.

13. References

13.1. Normative References

[CCITT.E163.1988] International Telephone and Telegraph Consultative Committee, "Numbering Plan for the International Telephone Service", CCITT Recommendation E.163, 1988.

[CCITT.X121.1988] International Telephone and Telegraph Consultative Committee, "International Numbering Plan for the Public Data Networks", CCITT Recommendation X.121, 1988.

[CCITT.X520.1988] International International Telephone and Telegraph Consultative Committee, "Information Technology - Open Systems Interconnection - The Directory: Selected Attribute Types", CCITT Recommendation X.520, November 1988.

[CCITT.X521.1988] International International Telephone and Telegraph Consultative Committee, "Information Technology - Open Systems Interconnection - The Directory: Selected Object Classes", CCITT Recommendation X.521, November 1988.

[I-D.mayrhofer-geo-uri] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Areas ('geo' URI)", draft-mayrhofer-geo-uri-02 (work in progress), February 2008.

[ISO.8601.2000] International Organization for Standardization, "Data elements and

interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, December 2000.

[ISO.8601.2004] International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, December 2004.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2425] Howes, T., Smith, M., and F. Dawson, "A MIME Content-Type for Directory Information", RFC 2425, September 1998.

[RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, September 1998.

[RFC2739] Small, T., Hennessy, D., and F. Dawson, "Calendar Attributes for vCard and LDAP", RFC 2739, January 2000.

[RFC2978] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier

(URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

[RFC4770] Jennings, C. and J. Reschke, Ed., "vCard Extensions for Instant Messaging (IM)", RFC 4770, January 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[oldreference_VCARD] Internet Mail Consortium, "vCard - The Electronic Business Card Version 2.1", September September.

13.2. Informative References

[ISO9070] The International Organization for Standardization, "ISO 9070, Information Processing - SGML support facilities - Registration Procedures for Public Text Owner Identifiers", April 1991.

[RFC3406] Daigle, L., van Gulik, D., Iannella, R., and P. Faltstrom, "Uniform Resource Names (URN) Namespace Definition Mechanisms", BCP 66, RFC 3406, October 2002.

URIs

[1] <mailto:vcard@ietf.org>

[2] <mailto:iana@iana.org>

Appendix A. Differences from RFCs 2425 and 2426

This appendix contains a list of changes that have been made in the vCard specification from RFCs 2425 and 2426.

A.1. New Structure

- [RFC2425] and [RFC2426] have been merged. Initially [RFC2425] was intended to be extensible but only 2426 ever extended it.
- vCard is now not only a MIME type but a stand-alone format.
- A proper MIME type registration form has been included.
- UTF-8 is now the default character set.
- New vCard elements can be registered from IANA.

A.2. Removed Features

- The group construct (i.e. GROUP.PROPERTY:...) no longer exists.
- The CONTEXT and CHARSET parameters are no more.
- The MAILER property is no more.
- The "intl", "dom", "postal", and "parcel" TYPE parameter values for the ADR and LABEL properties have been removed.
- Inline vCards (such as the value of the AGENT property) are no longer supported.
- In the N property, additional names are now subsumed into the given names list.

A.3. New Properties and Parameters

- The KIND, GENDER, LANG, DDAY, BIRTH, and DEATH properties have been added.
- [RFC2739], which defines the FBURL, CALADRURI, CAPURI, and CALURI properties, has been merged in.
- [RFC4770], which defines the IMPP property, has been merged in.
- The "work", "home", and "uri" TYPE parameter values for the EMAIL property have been added.

- The "pref" value of the TYPE parameter is now a parameter of its own, with a positive integer value indicating the level of preferredness.

A.4. Other Changes

- Synchronization is addressed in Section 8.
- The N property is no longer mandatory.
- The value of TEL is now a URI.
- The AGENT property was replaced with a type of RELATED.
- Date and time values now only support the basic format. Truncation is now supported.

Appendix B. Change Log (to be removed by RFC Editor prior to publication)

B.1. Changes in -07

- PREF is now bounded. 100 is the maximum value.
- Added the "emergency" RELATED type.
- Made GEO a URI.
- Added GEO and TZ parameters to ADR.
- Changed wording of "default" use of SOUND property.
- Completely reworked the date, time, and date-time grammars.
- Added the timestamp value type.
- REV now has the timestamp value type.
- Rewrote ABNF.
- ORG can now have a single level.

B.2. Changes in -06

- Corrected omission of resetability to text value for RELATED.
- Let KEY value type be reset to a URI value.

- ABNF fixes.
- Made gender values extensible.
- Gave the PREF parameter a positive integer value.
- Removed usage of the undefined "word" ABNF production rule.
- Defined property cardinalities.
- Defined properties allowable in WORK and HOME groups.
- Simplified the LANG property to use the vCard preference mechanism.
- Created the "language-tag" value type.
- Added PID to ABNF of SOURCE allowed parameters.
- Clarified escaping rules.
- Changed ABNF definition of non-standard X- properties.
- Removed TYPE parameter from EMAIL properties in examples.
- Created the CLIENTPIDMAP property.
- Changed PID value to a pair of small integers.
- Completely reworked synchronization mechanisms.
- Created brand new synchronization example.

B.3. Changes in -05

- Added multi PID value proposal.

B.4. Changes in -04

- Added "location" value for KIND property.
- Some fixes to ABNF.
- Moved "pref" from being a TYPE value to a parameter in its own right.
- Removed the "work" and "home" TYPE values.

- Reintroduced the group construct.
- Assigned meaning to WORK and HOME groups.
- Restricted the TEL TYPE parameter value set.
- In N property, removed additional names, and replaced with multiple given names.
- Removed TYPE parameter from EMAIL and IMPP properties.
- Replaced AGENT with a type of RELATED.
- Use example.org domain in URL example.
- Created initial IANA table of values.
- Defined meaning of PUBLIC, PRIVATE, CONFIDENTIAL.

B.5. Changes in -03

- Various changes to the synchronization mechanisms.
- Allowed truncated format for dated. See issue #236.

B.6. Changes in -02

- Removed useless text in IMPP description.
- Added CalDAV-SCHED example to CALADRURI.
- Removed CAPURI property.
- Dashes in dates and colons in times are now mandatory.
- Allow for dates such as 2008 and 2008-05 and times such as 07 and 07:54.
- Removed inline vCard value.
- Made AGENT only accept URI references instead of inline vCards.
- Added the MEMBER property.
- Renamed the UID parameter to PID.
- Changed the value type of the PID parameter to "a small integer."

- Changed the presence of UID and PID when synchronization is to be used from MUST to SHOULD.
- Added the RELATED (Section 7.6.6) property.
- Fixed many ABNF typos (issue #252).
- Changed formatting of ABNF comments to make them easier to read (issue #226).

B.7. Changes in -01

- Merged [RFC2739] in.
- Converted all foobar.com, abc.com, etc. to example.com.
- Fixed bugs in ABNF.
- Made explicit that coordinates in the GEO property are expressed in the WGS 84 reference system.
- Clarified folding issues with multi-byte characters.
- Made the value of TEL a URI.
- Added the UID parameter.
- Made the UID property's value type a URI.
- Added Section 8.
- Created IANA process for registering new parameters, value types, and properties.
- Created the initial IANA registries.
- Created vendor namespace based on text from RFC 4288.

B.8. Changes in -00

- Name change because draft has been accepted as WG item. Otherwise, same as draft-resnick-vcarddav-vcardrev-01.
- Removed reference to RFC 2234.
- Fixed errata from http://www.rfc-editor.org/errata_search.php/doc/html/rfc2426.

- Removed passage referring to RFC 2425 profiles.
- Renamed Section 7.4 from "Telecommunications Adressing Properties" to "Communications Properties.
- Added Appendix A and Appendix B.
- Added reference to [RFC4770].
- Removed the group construct.
- Made the N property no longer mandatory.
- Added the KIND property.
- Clarified meaning of TYPE parameter value for PHOTO, LOGO, KEY, and SOUND.
- Removed the CONTEXT parameter.
- Removed the MAILER property.
- Made reference to [ISO9070] informative.
- Removed "intl", "dom", "postal", and "parcel" TYPE parameter values for the ADR and LABEL properties.
- Clarified meaning of "extended address" ADR field.
- Mentioned [RFC3406] as another method of generating PRODID values.
- Updated obsolete references.
- Allowed BDAY and DDAY value types to be text values for fuzzy dates.
- Removed the CHARSET property. Now the encoding is always UTF-8, except when overridden by the Content-Type (which is considered a compatibility feature).

Authors' Addresses

Simon Perreault
Viagenie
2600 boul. Laurier, suite 625
Quebec, QC G1V 4W1
Canada

Phone: +1 418 656 9254
EMail: simon.perreault@viagenie.ca
URI: http://www.viagenie.ca

Peter W. Resnick
QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
US

Phone: +1 858 651 4478
EMail: presnick@qualcomm.com
URI: http://www.qualcomm.com/~presnick/