Network Working Group
Internet-Draft
Obsoletes: 2425, 2426, 4770 (if approved)
Updates: 2739 (if approved)
Intended status: Standards Track
Expires: April 17, 2010

S. Perreault
Viagenie

October 14, 2009

vCard XML Schema
draft-ietf-vcarddav-vcardxml-00

Status of This Memo

Copyright Notice

Abstract

This document defines the XML schema of the vCard data format.

Table of Contents

1. Introduction

vCard [I-D.ietf-vcarddav-vcardrev] is a data format for representing and exchanging information about individuals. It is a text-based format (as opposed to a binary format). This document defines an XML representation for vCard. The underlying data structure is exactly the same, enabling a 1-to-1 mapping between the original vCard format and the XML representation. The XML formatting may be preferred in some contexts where an XML engine is readily available and may be reused instead of writing a stand-alone vCard parser.

2. The Schema

The schema is expressed in the RELAX NG language [relaxng][relaxng-compact] and is found in Appendix A.

3. Example: Author's XML vCard

```
<?xml version="1.0" encoding="UTF-8"?>
<vcard xmlns="urn:ietf:params:xml:ns:vcard-4.0">
  <fn><text>Simon Perreault</text></fn>
  <n>
    <surname><text>Perreault</text></surname>
    <given><text>Simon</text></given>
    <prefix><text/></prefix>
    <suffix>
      <text>ing. jr.</text>
      <text>M.Sc.</text>
    </suffix>
  </n>
  <bday><date>--0203</date></bday>
  <anniversary>
    <date-time>20090808T1430-0500</date-time>
  </anniversary>
  <sex>1</sex>
  <lang><pref>1</pref><language-tag>fr</language-tag></lang>
  <lang><pref>2</pref><language-tag>en</language-tag></lang>
  <group name="work">
    <org><text>Viagenie</text></org>
    <adr>
      <pobox><text/></pobox>
      <ext><text>Suite 625</text></ext>
      <street><text>2600 boul. Laurier</text></street>
      <locality><text>Quebec</text></locality>
      <region><text>QC</text></region>
      <code><text>G1V 4W1</text></code>
      <country><text>Canada</text></country>
    </adr>
```

```
      <tel>
        <type><voice/></type>
        <uri>tel:+1-418-656-9254;ext=102</uri>
      </tel>
      <tel>
        <type><text/><voice/><cell/><video/></type>
        <uri>tel:+1-418-262-6501</uri>
      </tel>
      <tel>
        <type><fax/></type>
        <uri>tel:+1-418-656-9257</uri>
      </tel>
      <email><text>simon.perreault@viagenie.ca</text></email>
      <geo><uri>geo:46.772673,-71.282945</uri></geo>
      <key>
        <uri>http://www.viagenie.ca/simon.perreault/simon.asc</uri>
      </key>
    </group>
    <tz><text>America/Montreal</text></tz>
    <class><PUBLIC/></class>
  </vcard>
```

4. Design Considerations

The general idea is to map vCard parameters, properties, and value types to XML elements. For example, the "FN" property is mapped to the "fn" element. That element in turn contains a text element whose content corresponds to the vCard property's value.

vCard parameters are also mapped to XML elements. They are contained in property elements. For example, the "TYPE" parameter applied to the "TEL" property would look like the following in XML:

```
  <tel>
    <type><voice/></type>
    <uri>tel:+1-555-555-555</uri>
  </tel>
```

Properties having structured values (e.g. the N property) are expressed by XML element trees. Element names in that tree (e.g. "surname", "given", etc.) do not have a vCard equivalent since they are identified by position in plain vCard.

Line folding is a non-issue in XML. Therefore, the mapping from vCard to XML is done after the unfolding procedure is carried out. Conversely, the mapping from XML to vCard is done before the folding procedure is carried out.

4.1. Extensibility

The original vCard format is extensible. New properties, parameters, data types and values (collectively known as vCard objects) can be registered from IANA. It is expected that these vCard extensions will also specify extensions to the XML format described in this document. This is not a requirement: a separate document may be used instead.

Unregistered extensions (i.e. those starting with "X-" and "VND-...-") can be expressed in XML by making use of XML namespaces. They simply correspond to elements outside of the core namespace. For example:

```
  <ext:my-prop
      ext:xmlns="http://example.com/extensions/my-vcard">
    <pref>1</pref>               <!-- Core vCard elements  -->
    <text>value goes here</text>  <!-- are still accessible -->
  </ext:my-prop>
```

Note that extension elements do not need the "X- or "VND-" prefix in XML. The XML namespace mechanism is sufficient.

A vCard XML parser MUST ignore elements that are not part of this specification.

In the original vCard format, the VERSION property was mandatory and played a role in extensibility. In XML, this property is absent. Its role is played by the vCard core namespace identifier, which includes the version number. vCard revisions will use a different namespace.

Since vCard also has provisions for extending value enumerations, such as the allowed TYPE parameter values, these values are expressed using tags in XML.

4.2. Limitations

The schema does not validate the cardinality of properties. This is a limitation of the schema definition language. Cardinalities of the origiginal vCard format [I-D.ietf-vcarddav-vcardrev] MUST still be respected.

Some constructs (e.g. value enumerations in type parameters) have additional ordering constraints in XML. This is a result of limitations of the schema definition language and the order is arbitrary. The order MUST be respected in XML for the vCard to be valid. However, reordering as part of conversion to or from plain

vCard MAY happen.

5. Security Considerations

TBD

6. Normative References

[I-D.ietf-vcarddav-vcardrev] Perreault, S. and P. Resnick, "vCard Format Specification", draft-ietf-vcarddav-vcardrev-08 (work in progress), July 2009.

[relaxng] Clark, J., "RELAX NG Specification", December 2001.

[relaxng-compact] Clark, J., "RELAX NG Compact Syntax", November 2002.

Appendix A. Relax NG Schema

```
default namespace = "urn:ietf:params:xml:ns:vcard-4.0"

# Value types
\text = element text { text }
text-list = \text+
uri = element uri { xsd:anyURI }
date =
  element date {
    xsd:string { pattern = "\d{8}|\d{4}-\d\d|--\d\d(\d\d)?|---\d\d" }
  }
time =
  element time {
    xsd:string { pattern =
            "(\d\d(\d\d(\d\d)?)?|-\d\d(\d\d?)|--\d\d)"
          ~ "(Z|[+\-]\d\d(\d\d)?)?" }
  }
date-time =
  element date-time {
    xsd:string { pattern =
            "(\d{8}|--\d{4}|---\d\d)T\d\d(\d\d(\d\d)?)?"
          ~ "(Z|[+\-]\d\d(\d\d)?)?" }
  }
date-and-or-time = date | date-time | time
timestamp =
  element timestamp {
    xsd:string { pattern = "\d{8}T\d{6}(Z|[+\-]\d\d(\d\d)?)?" }
  }
```

```
   boolean = element boolean { xsd:boolean }
   integer = element integer { xsd:integer }
   float = element float { xsd:float }
   binary = element binary { xsd:base64Binary }
   language-tag =
     element language-tag {
       xsd:string { pattern =
               "([a-z]{2,3}((-[a-z]{3}){0,3})?|[a-z]{4,8})"
             ~ "(-[a-z]{4})?(-([a-z]{2}|\d{3}))?"
             ~ "(-([0-9a-z]{5,8}|\d[0-9a-z]{3}))*"
             ~ "(-[0-9a-wyz](-[0-9a-z]{2,8})+)*(-x(-[0-9a-z]{1,8})+)?|"
             ~ "x(-[0-9a-z]{1,8})+|[a-z]{1,3}(-[0-9a-z]{2,8}){1,2}" }
     }
   data =
     (element type {
        xsd:string { pattern = "[A-Za-z\d!#$&.+\-^_]{1,127}/"
                             ~ "[A-Za-z\d!#$&.+\-^_]{1,127}" }
      },
      binary)
     | uri

   # Parameters
   language = element language { language-tag }?
   pref =
     element pref {
       xsd:integer { minInclusive = "1" maxInclusive = "100" }
     }?
   pid =
     element pid {
       xsd:string { pattern = "\d+(\.\d+)?" }
     }?

   # Properties
   source = element source { pid, pref, uri }
   name = element name { \text }
   kind =
     element kind {
       element individual { empty }
       | element group { empty }
       | element org { empty }
       | element location { empty }
     }
   fn = element fn { language, pid, pref, \text }
   n =
     element n {
       language,
       element surname { text-list },
       element given { text-list },
```

```
      element prefix { text-list },
      element suffix { text-list }
    }
  nickname = element nickname { language, pid, pref, text-list }
  photo = element photo { pid, pref, data }
  bday = element bday { date-and-or-time | \text }
  dday = element dday { date-and-or-time | \text }
  birth = element birth { language, \text }
  death = element death { language, \text }
  anniversary = element anniversary { date-and-or-time | \text }
  sex = element sex { "0" | "1" | "2" | "9" }
  adr =
    element adr {
      language,
      element geo { uri }?,
      element tz { \text | uri }?,
      pid,
      pref,
      element pobox { text-list },
      element ext { text-list },
      element street { text-list },
      element locality { text-list },
      element region { text-list },
      element code { text-list },
      element country { text-list }
    }
  label = element label { language, pid, pref, \text }
  tel =
    element tel {
      element type {
        element text { empty }?,
        element voice { empty }?,
        element fax { empty }?,
        element cell { empty }?,
        element video { empty }?,
        element pager { empty }?
      }?,
      pid,
      pref,
      uri
    }
  email = element email { pid, pref, \text }
  impp = element impp { pid, pref, uri }
  lang = element lang { pid, pref, language-tag }
  tz = element tz { pid, pref, (\text | uri) }
  geo = element geo { pid, pref, uri }
  title = element title { language, pid, pref, \text }
  role = element role { language, pid, pref, \text }
```

```
   logo = element logo { language, pid, pref, data }
   org = element org { language, pid, pref, text-list }
   member = element member { pid, pref, uri }
   related =
     element related {
       element type {
         element parent { empty }
         | element child { empty }
         | element sibling { empty }
         | element spouse { empty }
         | element family { empty }
         | element friend { empty }
         | element supervisor { empty }
         | element supervisee { empty }
         | element assistant { empty }
         | element colleague { empty }
         | element agent { empty }
         | element emergency { empty }
       }?,
       pid,
       pref,
       (uri | \text)
     }
   categories = element categories { pid, pref, \text }
   note = element note { language, pid, pref, \text }
   prodid = element prodid { \text }
   rev = element rev { timestamp }
   sort-string = element sort-string { \text }
   sound = element sound { language, pid, pref, data }
   uid = element uid { uri }
   clientpidmap =
     element clientpidmap {
       element sourceid { xsd:positiveInteger },
       uri
     }
   url = element url { pid, pref, uri }
   class =
     element class {
       element PUBLIC { empty }
       | element PRIVATE { empty }
       | element CONFIDENTIAL { empty }
     }
   key = element key { pid, pref, data }
   fburl = element fburl { pid, pref, uri }
   caladruri = element caladruri { pid, pref, uri }
   caluri = element caluri { pid, pref, uri }

   # Top-level grammar
```

```
   group-property = fn | nickname | photo | adr | label | tel | email |
     impp | lang | tz | geo | title | role | logo | org | related |
     categories | note | sound | url | key | fburl | caladruri | caluri
   any-property = group-property | source | name | kind | n | bday |
     dday | birth | death | anniversary | sex | member | prodid |
     rev | sort-string | uid | clientpidmap | class
   start =
     element vcard {
       (any-property
        | element group {
            attribute name { text },
            group-property*
          })*
     }
```

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

B.1. Changes in -00

- Same as draft-perreault-vcarddav-vcardxml-02.

Author's Address

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