

Network Working Group
Perreault
Internet-Draft
Viagenie
Intended status: Standards Track
2010
Expires: January 13, 2011

S.

July 12,

vCard XML Representation
draft-ietf-vcarddav-vcardxml-04

Abstract

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

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 13, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

Perreault
1]

Expires January 13, 2011

[Page

the Trust Legal Provisions and are provided without warranty as described in the BSD License.

Table of Contents

1.	Introduction	
3		
2.	The Schema	
3		
3.	Example: Author's XML vCard	
3		
4.	Design Considerations	
5		
4.1.	Extensibility	
6		
4.2.	Limitations	
7		
5.	Format Conversions	
8		
6.	Security Considerations	
10		
7.	IANA Considerations	
10		
7.1.	Registration of the XML Namespace	
10		
7.2.	Media Type	
10		
8.	Acknowledgements	
11		
9.	References	
11		
9.1.	Normative References	
11		
9.2.	Informative References	
11		
Appendix A.	Relax NG Schema	
12		
Appendix B.	Change Log (to be removed by RFC Editor prior to publication)	
18		
B.1.	Changes in -04	
18		
B.2.	Changes in -03	
18		
B.3.	Changes in -02	
18		
B.4.	Changes in -01	
18		
B.5.	Changes in -00	
19		

Perreault
2]

Expires January 13, 2011

[Page

1. Introduction

vCard [[I-D.ietf-vcarddav-vcardrev](#)] is a data format for representing and exchanging information about individuals and other entities. 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.

Earlier work on an XML format for vCard was started in 1998 by Frank Dawson [[I-D.dawson-vcard-xml-dtd](#)]. Sadly it did not take over the world.

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"?>
<vcards xmlns="urn:ietf:params:xml:ns:vcard-4.0">
  <vcard>
    <fn><text>Simon Perreault</text></fn>
    <n>
      <surname><text>Perreault</text></surname>
      <given><text>Simon</text></given>
      <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><integer>1</integer></sex>
    <lang>
      <parameters><pref>1</pref></parameters>
      <language-tag>fr</language-tag>
    </lang>
    <lang>
      <parameters><pref>2</pref></parameters>
      <language-tag>en</language-tag>
    </lang>
  </vcard>
</vcards>
```



```
</lang>
<org>
  <parameters><type>work</type></parameters>
  <text>Viagenie</text>
</org>
<adr>
  <parameters><type>work</type></parameters>
  <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>
  <parameters>
    <type>work</type>
    <type>voice</type>
  </parameters>
  <uri>tel:+1-418-656-9254;ext=102</uri>
</tel>
<tel>
  <parameters>
    <type>work</type>
    <type>text</type>
    <type>voice</type>
    <type>cell</type>
    <type>video</type>
  </parameters>
  <uri>tel:+1-418-262-6501</uri>
</tel>
<tel>
  <parameters>
    <type>work</type>
    <type>fax</type>
  </parameters>
  <uri>tel:+1-418-656-9257</uri>
</tel>
<email>
  <parameters><type>work</type></parameters>
  <text>simon.perreault@viagenie.ca</text>
</email>
<geo>
  <parameters><type>work</type></parameters>
  <uri>geo:46.772673, -71.282945</uri>
</geo>
<key>
```



```
    <parameters><type>work</type></parameters>
    <uri>http://www.viagenie.ca/simon.perreault/simon.asc</uri>
  </key>
  <tz><text>America/Montreal</text></tz>
  <class><text>PUBLIC</text></class>
</vcard>
</vcards>
```

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 the <parameters> element, which is contained in property elements.

For example, the "TYPE" parameter applied to the "TEL" property would

look like the following in XML:

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

Parameters taking a list of values are simply repeated multiple times, once for each value in the list.

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.

A top-level <vcards> element is used as root. It contains one or more <vcard> element, each representing a complete vCard. The <vcards> element MUST be present even when only a single vCard is present in an XML document.

The group construct (Section 3.2 in [[I-D.ietf-vcarddav-vcardrev](#)]) is

Perreault
5]

Expires January 13, 2011

[Page

represented with the <group> element. The "name" attribute contains the group's name. For example:

```
<vcards>
  <vcard>
    <group name="contact">
      <fn>...</fn>
      <email>...</email>
    </group>
    <group name="media">
      <photo>...</photo>
    </group>
    <categories>...</categories>
  </vcard>
</vcards>
```

is equivalent to:

```
BEGIN:VCARD
VERSION:4.0
contact.FN=...
contact.EMAIL=...
media.PHOTO=...
CATEGORIES=...
END:VCARD
```

4.1. Extensibility

The original vCard format is extensible. New properties, parameters, data types and values (collectively known as vCard objects) can be registered with IANA. It is expected that these vCard extensions will also specify extensions to the XML format described in this document.

Unregistered extensions (i.e. those starting with "X-" and "VND-...-") are expressed in XML by using elements starting with "x-" and "vnd-...-". Usage of XML namespaces for extensibility is RECOMMENDED for extensions that have no equivalent in plain text vCard. Refer to [Section 5](#) for the implications when converting between plain-text vCard and XML.

Examples:

```
<x-my-prop>
  <parameters>
    <pref>1</pref>
    <text>value goes here</text>
</x-my-prop>

<ext:my-prop
  ext:xmlns="http://example.com/extensions/my-vcard">
  <parameters>
    <pref>1</pref>
  </parameters>
  <text>value goes here</text>
</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 XML elements and attributes for which it doesn't recognize the expanded name. The normal behaviour of ignoring XML processing instructions whose target is not recognized MUST also be followed.

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.

Parameters containing a list of values are expressed using a list of elements in XML (e.g. the <type> element).

4.2. Limitations

The schema does not validate the cardinality of properties. This is a limitation of the schema definition language. Cardinalities of the

original 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.

Perreault
7]

Expires January 13, 2011

[Page

5. Format Conversions

When converting from XML vCard (this specification) to plain-text vCard [[I-D.ietf-vcarddav-vcardrev](#)], the following rules apply:

- o Properties in the vCard 4 namespace:
 - * If the converter knows of a specific plain-text representation for this property, it uses it. For example, the <adr> element corresponds to the "ADR" property, which is encoded using comma-separated lists separated by semi-colons.
 - * Otherwise, the property name is taken from the element name, property parameters are taken from the <parameters> element, and the content of the property is taken from the content of the value element. If the property element has attributes or contains other XML elements, they are dropped.
 - * If a standard property's XML element contains XML elements and attributes for which the converter doesn't recognize the expanded name, they are dropped. Therefore, it is RECOMMENDED to limit extensions to the property level to ensure that all data is preserved intact in round-trip conversions.
- o Properties in other namespaces are wrapped as-is inside an "XML" property.
- o Property value escaping (Section 3.3 of [[I-D.ietf-vcarddav-vcardrev](#)]) is carried out. For example, a NEWLINE character (ASCII decimal 10) becomes "\n".
- o Double-quoting of parameter values, as well as backslash escaping in parameter values, is carried out. For example, <param>"foo", "bar"</param> becomes PARAM="\ "foo\ ", \ "bar\ "".

When converting from plain-text vCard [[I-D.ietf-vcarddav-vcardrev](#)] to XML vCard (this specification), the following rules apply:

- o The content of "XML" properties is converted as-is to XML.
- o Properties for which the converter knows of a specific XML representation use it. For example, the "ADR" property is represented using the <adr> element and related sub-elements.
- o Other properties are converted to XML in the following way:
 - * The XML namespace of the property element is set to the vCard namespace.

- * The name of the property element is set to the lowercased name of the property.
 - * If the property has parameters, they get translated as-is (without lowercasing of parameter names, removal of backslash escaping, and removal of quoting) into sub-elements of the <parameters> element
 - * The property element contains a single <text> element whose content is copied as-is from the property's value.
- o Property value escaping is undone. For example, "\n" becomes a NEWLINE character (ASCII decimal 10).
 - o Double-quoting of parameter values, as well as backslash escaping in parameter values, is undone. For example, PARAM="\\"foo\\",\\"bar\\" becomes <param>"foo","bar"</param>.

For example, these two vCards are equivalent:

```
<xml version="1.0">
  <vcards xmlns="urn:ietf:params:xml:ns:vcard-4.0">
    <vcard>
      <fn><text>J. Doe</text></fn>
      <n>
        <surname><text>Doe</text></surname>
        <given><text>J.</text></given>
        <prefix/>
        <suffix/>
      </n>
      <x-file>
        <parameters><type>image/jpeg</type></parameters>
        <text>alien.jpg</text>
      </x-file>
      <a xmlns="http://www.w3.org/1999/xhtml"
        href="http://www.example.com">My web page!</a>
    </vcard>
  </vcards>
</xml>
```

```
BEGIN:VCARD
VERSION:4.0
FN:J. Doe
N:Doe;J.;
X-FILE;TYPE=image/jpeg:alien.jpg
XML:<a xmlns="http://www.w3.org/1999/xhtml"\n
  href="http://www.example.com">My web page!</a>
```


END:VCARD

6. Security Considerations

All the security considerations applicable to plain vCard [[I-D.ietf-vcarddav-vcardrev](#)] are applicable to this document as well.

7. IANA Considerations

7.1. Registration of the XML Namespace

URI: urn:ietf:params:xml:ns:vcard-4.0

Registrant Contact: Simon Perreault <simon.perreault@viagenie.ca>

XML: None. Namespace URIs do not represent an XML specification.

7.2. Media Type

This section defines the MIME media type for use with vCard-in-XML data.

To: ietf-types@iana.org

Subject: Registration of media type application/vcard+xml

Type name: application

Subtype name: vcard+xml

Required parameters: none

Optional parameters: none

Encoding considerations: Same as for application/xml.

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

Interoperability considerations: This media type provides an alternative syntax to vCard data [[I-D.ietf-vcarddav-vcardrev](#)] based on XML.

Published specification: This specification.

Applications which use this media type: Applications that currently make use of the text/vcard media type can use this as an alternative.

Additional information:

Magic number(s): none

File extension(s): XML data should use ".xml" as the file extension.

Macintosh file type code(s): none

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

Intended usage: COMMON

Restrictions on usage: none

Author: Simon Perreault

Change controller: IETF

8. Acknowledgements

Thanks to the following people for their input:

Alexey Melnikov, Barry Leiba, Cyrus Daboo, Joe Hildebrand, Joseph Smarr, Marc Blanchet, Peter Saint-Andre, Robins George, Zahhar Kirillov, Zoltan Ordogh.

9. References

9.1. Normative References

[I-D.ietf-vcarddav-vcardrev] Perreault, S. and P. Resnick, "vCard Format Specification", [draft-ietf-vcarddav-vcardrev-12](#) (work in progress), July 2010.

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

[relaxng-compact] Clark, J., "RELAX NG Compact Syntax", November 2002, <<http://www.relaxng.org/compact-20021121.html>>.

9.2. Informative References

[I-D.dawson-vcard-xml-dtd] Dawson, F., "The vCard v3.0 XML DTD", [draft-dawson-vcard-xml-dtd-03](#) (work in

progress), June 1998.

[Appendix A.](#) Relax NG Schema

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

# Value types
value-text = element text { text }
value-text-list = value-text+
value-uri = element uri { xsd:anyURI }
value-date = element date {
  xsd:string { pattern = "\d{8}|\d{4}-\d\d|-\d\d(\d\d)?|---\d\d" }
}
value-time = element time {
  xsd:string { pattern = "(\d\d(\d\d(\d\d)?)?|-\d\d(\d\d?)|-\d\d)"
    ~ "(Z|[+|-]\d\d(\d\d)?)" }
}
value-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)?)" }
}
value-date-and-or-time = value-date | value-date-time | value-time
value-timestamp = element timestamp {
  xsd:string { pattern = "\d{8}T\d{6}(Z|[+|-]\d\d(\d\d)?)" }
}
value-boolean = element boolean { xsd:boolean }
value-integer = element integer { xsd:integer }
value-float = element float { xsd:float }
value-binary = element binary { xsd:base64Binary }
value-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}" }
}

# Parameters
param-language = element language { value-language-tag }?
param-pref = element pref {
  xsd:integer { minInclusive = "1" maxInclusive = "100" }
}?
param-pid = element pid {
  xsd:string { pattern = "\d+(\.\d+)" }
}?
param-type = element type { "work" | "home" }*
param-calscale = element calscale { "gregorian" }?
```



```
param-sort-as = element sort-as { value-text+ }?  
param-version = element version { value-text }?  
param-geo = element geo { value-uri }?  
param-tz = element tz { value-text | value-uri }?  
param-fmttype = element fmttype {  
    xsd:string { pattern = "[a-zA-Z0-9!#$%&.\-^_]{1,127}/"  
        ~ "[a-zA-Z0-9!#$%&.\-^_]{1,127}" }  
    }?
```

Properties

```
property-source = element source {  
    element parameters { param-pid, param-pref },  
    value-uri  
}  
property-name = element name { value-text }  
property-kind = element kind {  
    element text { "individual" | "group" | "org" |  
        "location" | "thing" }*  
}  
property-fn = element fn {  
    element parameters { param-language, param-pid,  
        param-pref, param-type }?,  
    value-text  
}  
property-n = element n {  
    element parameters { param-language, param-sort-as }?,  
    element surname { value-text-list? },  
    element given { value-text-list? },  
    element prefix { value-text-list? },  
    element suffix { value-text-list? }  
}  
property-nickname = element nickname {  
    element parameters { param-language, param-pid,  
        param-pref, param-type }?,  
    value-text-list  
}  
property-photo = element photo {  
    element parameters {  
        param-pid,  
        param-pref,  
        param-type,  
        param-fmttype  
    }?,  
    (value-binary | value-uri)  
}  
property-bday = element bday {  
    element parameters { param-calscale }?,  
    (value-date-and-or-time | value-text)
```



```
    }
property-dday = element dday {
    element parameters { param-calscale }?,
    (value-date-and-or-time | value-text)
}
property-birth = element birth {
    element parameters { param-language }?,
    (value-text | value-uri)
}
property-death = element death {
    element parameters { param-language }?,
    (value-text | value-uri)
}
property-anniversary = element anniversary {
    element parameters { param-calscale }?,
    (value-date-and-or-time | value-text)
}
property-sex = element sex {
    element integer { "0" | "1" | "2" | "9" }
}
property-adr = element adr {
    element parameters {
        param-language,
        param-pid,
        param-pref,
        param-type,
        param-geo,
        param-tz
    }?,
    element pobox { value-text-list? },
    element ext { value-text-list? },
    element street { value-text-list? },
    element locality { value-text-list? },
    element region { value-text-list? },
    element code { value-text-list? },
    element country { value-text-list? }
}
property-label = element label {
    element parameters { param-language, param-pid,
        param-pref, param-type }?,
    value-text
}
property-tel = element tel {
    element parameters {
        param-pid,
        param-pref,
        element type { "work" | "home" | "text" | "voice"
            | "fax" | "cell" | "video" | "pager"
        }
    }
}
```



```
        | "textphone" }*
    },
    value-uri
}
property-email = element email {
    element parameters { param-pid, param-pref, param-type }?,
    value-text
}
property-impp = element impp {
    element parameters { param-pid, param-pref, param-type }?,
    value-uri
}
property-lang = element lang {
    element parameters { param-pid, param-pref, param-type }?,
    value-language-tag
}
property-tz = element tz {
    element parameters { param-pid, param-pref, param-type }?,
    (value-text | value-uri)
}
property-geo = element geo {
    element parameters { param-pid, param-pref, param-type }?,
    value-uri
}
property-title = element title {
    element parameters { param-language, param-pid,
        param-pref, param-type }?,
    value-text
}
property-role = element role {
    element parameters { param-language, param-pid,
        param-pref, param-type }?,
    value-text
}
property-logo = element logo {
    element parameters {
        param-language,
        param-pid,
        param-pref,
        param-type,
        param-fmttype
    }?,
    (value-binary | value-uri)
}
property-org = element org {
    element parameters { param-language, param-pid,
        param-pref, param-type, param-sort-as }?,
    value-text-list
}
```



```
    }
property-member = element member { param-pid, param-pref, value-uri }
property-related = element related {
    element parameters {
        param-pid,
        param-pref,
        element type { "work" | "home" | "parent" | "child"
            | "sibling" | "spouse" | "family" | "friend"
            | "supervisor" | "supervisee" | "assistant"
            | "colleague" | "agent" | "emergency" }*
    }?,
    (value-uri | value-text)
}
property-categories = element categories {
    element parameters { param-pid, param-pref, param-type }?,
    value-text
}
property-note = element note {
    element parameters { param-language, param-pid,
        param-pref, param-type }?,
    value-text
}
property-prodid = element prodid { value-text }
property-rev = element rev { value-timestamp }
property-sound = element sound {
    element parameters {
        param-language,
        param-pid,
        param-pref,
        param-type,
        param-fmttype
    }?,
    (value-binary | value-uri)
}
property-uid = element uid { value-uri }
property-clientpidmap = element clientpidmap {
    element sourceid { xsd:positiveInteger },
    value-uri
}
property-url = element url {
    element parameters { param-pid, param-pref, param-type }?,
    value-uri
}
property-class = element class {
    element text { "PUBLIC" | "PRIVATE" | "CONFIDENTIAL" }
}
property-key = element key {
    element parameters {
```



```
        param-pid,  
        param-pref,  
        param-type,  
        param-fmttype  
    }?,  
    (value-binary | value-uri | value-text)  
}  
property-fburl = element fburl {  
    element parameters { param-pid, param-pref, param-type }?,  
    value-uri  
}  
property-caladruri = element caladruri {  
    element parameters { param-pid, param-pref, param-type }?,  
    value-uri  
}  
property-caluri = element caluri {  
    element parameters { param-pid, param-pref, param-type }?,  
    value-uri  
}  
  
# Top-level grammar  
property = property-adr | property-anniversary | property-bday  
    | property-birth | property-caladruri | property-caluri  
    | property-categories | property-class | property-clientpidmap  
    | property-dday | property-death | property-email  
    | property-fburl | property-fn | property-geo | property-impp  
    | property-key | property-kind | property-label | property-  
lang  
    | property-logo | property-member | property-n | property-name  
    | property-nickname | property-note | property-org  
    | property-photo | property-prodid | property-related  
    | property-rev | property-role | property-sex  
    | property-sound | property-source | property-tel  
    | property-title | property-tz | property-uid | property-url  
start = element vcards {  
    element vcard {  
        (property  
        | element group {  
            attribute name { text },  
            property*  
        })+  
    }+  
}
```


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

B.1. Changes in -04

- o Synchronized with [draft-ietf-vcarddav-vcardrev-12](#).
- o Added application/vcard+xml media type.
- o Added rules for backslash escaping and quoting when converting.
- o Added description of <vcards> element.
- o Described group construct in XML.

B.2. Changes in -03

- o Created "Format Conversions" section.
- o Turned more <type> parameter values into plain text.
- o Removed need for empty value elements in components.
- o Wrapped value of <sex>, <class>, and <kind> in value elements.

B.3. Changes in -02

- o Synchronized with [draft-ietf-vcarddav-vcardrev-10](#).
- o Turned <type> parameter values into plain text.
- o Moved the "XML" property to vCard base.
- o Changed title to avoid confusion with XML Schema.
- o Added prefixes "value-", "param-", and "property-" in schema.
- o Better language for specifying what a parser must ignore.

B.4. Changes in -01

- o Synchronized with [draft-ietf-vcarddav-vcardrev-09](#).
- o Added the <vcards> element to allow multiple vCards in a single XML file.
- o Created the <parameters> container element.

- o Use text value for enumeration in <class> element.
- o Created the "XML" vCard property.
- o Added IANA considerations section.
- o Added security considerations section.

B.5. Changes in -00

- o 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>

