This document specifies an XMPP extension for use of the vCard4 XML format in XMPP systems, with the intent of obsoleting the vcard-temp format.
Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2018 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

## NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. ##

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3.5</td>
<td>TEL</td>
<td>16</td>
</tr>
<tr>
<td>11.4</td>
<td>Properties Defined Similarly in vcard-temp, vCard3, and vCard4</td>
<td>16</td>
</tr>
<tr>
<td>11.5</td>
<td>Properties Defined in vCard3 but Removed from vCard4</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>Migration Tools</td>
<td>18</td>
</tr>
<tr>
<td>12.1</td>
<td>Extensible Stylesheet Language Transformation (XSLT) from vcard-temp to vCard4</td>
<td>18</td>
</tr>
<tr>
<td>12.2</td>
<td>Example of vcard-temp Data</td>
<td>29</td>
</tr>
<tr>
<td>12.3</td>
<td>Example of vCard4 XML Data</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>Acknowledgements</td>
<td>41</td>
</tr>
</tbody>
</table>
1 Introduction

Since 1999, the Jabber/XMPP community has used an interim, unofficial XML representation of vCard data for personal contacts, called vcard-temp (XEP-0054) \(^1\). Recently, the IETF has upgraded vCard from vCard3 to vCard 4 (RFC 6350 \(^2\)), and at the same time has defined an official XML format for vCard4 (RFC 6351 \(^3\)). This document specifies an XMPP extension for use of the vCard4 XML format in XMPP systems, with the intent of obsoleting the vcard-temp format. Primarily this document defines the encapsulation method itself; secondarily it also defines transport methods and a mapping to the vcard-temp format for migration by clients and servers.

2 Requirements

This specification was designed with the following requirements in mind.

1. Reuse vCard4 as defined in RFC 6350.
2. Reuse the vCard4 XML format as defined in RFC 6351.
3. Ensure that clients and servers can easily migrate from vcard-temp to the new encapsulation format.
4. Notify interested parties of changes to vCard data using standard XMPP extensions, specifically Personal Eventing Protocol (XEP-0163) \(^4\).
5. Support vCards for non-human entities such as XMPP servers and Multi-User Chat (XEP-0045) \(^5\) rooms.

3 Reuse of vCard4

Because there is now an XML namespace for the official vCard format, we can simply re-use that namespace: "urn:ietf:params:xml:ns:vcard-4.0". The vCard XML format defined at the IETF specifies that the root element is `<vcard/>`, where the only defined child element is `<vcard/>`. For use in XMPP, we specify that the root element shall be `<vcard/>`, not `<vcards/>`.

---

4 Self vCards

This section describes the use of the vCard format for self-publication and retrieval of publicly-accessible information about any entity on an XMPP network, thus fulfilling all the use cases of the old vcard-temp format.

4.1 IQ-Based Publication and Retrieval

As in XEP-0054, the primary method for publishing and retrieving vCards is the XMPP <iq/> stanza. (Although it would have been possible to use Best Practices for Persistent Storage of Public Data via Publish-Subscribe (XEP-0222) for public storage and retrieval, community consensus is that storage via IQ is more backward-compatible with XEP-0054, and that publish-subscribe is more appropriate only for event notifications.)

4.1.1 Retrieval

An XMPP entity retrieves the vCard of another entity (or itself) by sending an IQ-get to the target entity containing a <vcard/> child element (note the lowercase "c"!) qualified by the 'urn:ietf:params:xml:ns:vcard-4.0' namespace.

Listing 1: vCard Retrieval Request

```
<iq from='samizzi@cisco.com/foo'
    id='bx81v356'
    to='stpeter@jabber.org'
    type='get'>
    <vcard xmlns='urn:ietf:params:xml:ns:vcard-4.0'/>
</iq>
```

If a vCard exists for the target entity, the responsible entity (e.g., the XMPP server that hosts the account for a bare JID) MUST return the data in an IQ-result:

Listing 2: Server Returns vCard

```
<iq from='stpeter@jabber.org'
    id='bx81v356'
    to='samizzi@cisco.com/foo'
    type='result'>
    <vcard xmlns='urn:ietf:params:xml:ns:vcard-4.0'>
      <fn><text>Peter Saint-Andre</text></fn>
      <n><surname>Saint-Andre</surname><given>Peter</given><additional></additional></n>
</vcard>
</iq>
```

---


2
<nickname><text>stpeter</text></nickname>
<nickname><text>psa</text></nickname>
<photo><uri>https://stpeter.im/images/stpeter_oscon.jpg</uri></photo>
bday><date>1966-08-06</date></bday>
<adr>
  <parameters>
    <type><text>work</text><text>voice</text></type>
    <pref><integer>1</integer></pref>
  </parameters>
  <ext>Suite 600</ext>
  <street>1899 Wynkoop Street</street>
  <locality>Denver</locality>
  <region>CO</region>
  <code>80202</code>
  <country>USA</country>
</adr>
<adr>
  <parameters><type><text>home</text></type></parameters>
  <ext>Parker</ext>
  <locality>Parker</locality>
  <region>CO</region>
  <code>80138</code>
  <country>USA</country>
</adr>
tel>
  <parameters>
    <type><text>work</text><text>voice</text></type>
    <pref><integer>1</integer></pref>
  </parameters>
  <uri>tel: +1-303-308-3282</uri>
</tel>
tel>
  <parameters><type><text>work</text><text>fax</text></type></parameters>
  <uri>tel: +1-303-308-3219</uri>
</tel>
tel>
  <parameters>
    <type><text>cell</text><text>voice</text><text>text</text></type></parameters>
  <uri>tel: +1-720-256-6756</uri>
</tel>
tel>
  <parameters><type><text>home</text><text>voice</text></type></parameters>
  <uri>tel: +1-303-555-1212</uri>
</tel>
<tel>
<geo>geo:39.59,-105.01</geo>
<title>Executive Director</title>
<role>Patron Saint</role>
<org><parameters><type>work</type><text>XMPP Standards Foundation</text></parameters></org>
<url>https://stpeter.im/</url>
<note>More information about me is located on my personal website: https://stpeter.im/</note>
<gender><sex>M</sex></gender>
<lang><parameters><pref>1</pref></parameters></lang>
<email><parameters><type>work</type><text>psaintan@cisco.com</text></parameters></email>
<email><parameters><type>home</type><text>stpeter@jabber.org</text></parameters></email>
<impp><parameters><type>work</type><uri>xmpp:psaintan@cisco.com</uri></parameters></impp>
<impp><parameters><type>home</type><uri>xmpp:stpeter@jabber.org</uri></parameters></impp>
?key><uri>https://stpeter.im/stpeter.asc</uri></key>
</vcard>
</iq>

If no vCard exists, the server MUST return an IQ-result containing an empty <vcard/> element.

Listing 3: No vCard (empty element)

<iq from='stpeter@jabber.org'
    id='bx81v356'
    to='samizzi@cisco.com/foo'>
4.1.2 Publication

An XMPP entity publishes or updates its vCard by sending an IQ-set to itself (typically its bare JID), containing a `<vcard/>` child element qualified by the ‘urn:ietf:params:xml:ns:vcard-4.0’ namespace. The publication request needs to include the entire vCard, not a “diff” against the prior data (if any).

Listing 4: vCard Publication Request

```
<iq from='stpeter@jabber.org/squire
     id='h3vz319m'
     to='stpeter@jabber.org'
     type='set'>
  <vcard xmlns='urn:ietf:params:xml:ns:vcard-4.0'>
    [...]
  </vcard>
</iq>
```

If no error occurs, the responsible entity returns an IQ-result.

Listing 5: Server Acknowledges Publication

```
<iq from='stpeter@jabber.org'
     id='bx81v356'
     to='stpeter@jabber.org/squire'
     type='result'/>
```

Note: An entity MAY have authorization to update the vCard of another entity (e.g., a server administrator might have authorization to modify the server’s vCard).

4.2 Event Notifications

Publish-Subscribe (XEP-0060) provides a way to subscribe to events, and Personal Eventing Protocol (XEP-0163) defines a pubsub profile for events associated with instant messaging (IM) accounts. If PEP is supported by an IM server, it can be used to automatically generate event notifications when a user’s vCard is modified.

---

4.2.1 Location

The canonical location for notifications regarding a user’s vCard is a pubsub node whose name is “urn:xmpp:vcard4”.

4.2.2 Subscribing to vCard Notifications

Let us imagine that Juliet wishes to receive the updates that Romeo publishes to his vCard. She has two options:

1. Implicitly subscribe by advertising support for "urn:xmpp:vcard4+notify" in her Entity Capabilities (XEP-0115) data. Romeo’s PEP service then automatically sends vCard updates to her when it receives presence from her, until and unless she sends presence of type unavailable or stops advertising an interest in vCard updates. This is in accordance with XEP-0060, section 6.1.

2. Explicitly subscribe by sending a formal subscription request to the "urn:xmpp:vcard4" node at Romeo’s JabberID. Romeo’s PEP service might send her all vCard updates even if she is offline at the time (depending on service policies regarding presence integration).

4.2.3 Receiving a vCard Notification

Because Juliet has sent presence to Romeo including Entity Capabilities data that includes the "urn:xmpp:vcard4+notify" feature, Romeo’s XMPP server will send a PEP notification to Juliet. The notification can include an XMPP message body for backward-compatibility with XMPP clients that are not pubsub-capable. This is in accordance with XEP-0060, second 6.1.7.

Listing 6: Receiving a vCard publication/update

```xml
<message from='romeo@montague.lit'
to='juliet@capulet.lit'
type='headline'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:xmpp:vcard4'>
      <item id='current'/>
    </items>
  </event>
</message>
```

Note: There is no payload, because this is a pure notification (the receiver needs to retrieve the vCard using an IQ-get as described earlier).

---

5 Contact vCards

In addition to enabling the publication and retrieval of vCards about any entity on an XMPP network, the vCard format can also be used to store information about an entity’s contacts.

5.1 Format

A contact is simply a vCard about someone else (or something else, in the case of automated entities). If the other person or entity is in the user’s roster, the vCard SHOULD contain the Jabber ID of the person or entity. This enables a user to store information about the contact outside of the roster, thus obviating the need for changes or extensions to the roster namespace itself (as in Annotations (XEP-0145)\(^ {11} \)).

Listing 7: Contact

| vcard xmlns="urn:ietf:params:xml:ns:vcard-4.0" |
| fn>text>Samantha Mizzi</text> |
| n> |
| surname>Mizzi</surname> |
| given>Samantha</given> |
| additional> |
| nickname>text>Sam</text> |
| nickname>text>samizzi</text> |
| geo>uri>geo:39.59,-105.01</uri> |
| org> |
| parameters>type>text>work</text> |
| text>Cisco</text> |
| note> |
| text>My co-author on XEP-0292. She’s cool!</text> |
| impp> |
| parameters>type>text>work</text> |
| uri>xmpp:samizzi@cisco.com</uri> |


5.2 Storage

Because contact vCards are private information, they are best stored using Best Practices for Persistent Storage of Private Data via Publish-Subscribe (XEP-0223)\(^\text{12}\). The canonical location is a well-known pubsub node "urn:xmpp:contacts". In accordance with XEP-0223, this node MUST have an access type of "whitelist" by default. When a client stores items at this node, it SHOULD NOT include an ItemID, so that the pubsub service can assign those identifiers.

Listing 8: Storing a Contact vCard

```
<iq from='stpeter@stpeter.im/squire' type='set' id='h3vs7163'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:contacts'>
      <item>
        <vcard xmlns='urn:ietf:params:xml:ns:vcard-4.0'>
          <fn><text>Samantha Mizzi</text></fn>
          <n>
            <surname>Mizzi</surname>
            <given>Samantha</given>
            <additional></additional>
          </n>
          <nickname><text>Sam</text></nickname>
          <nickname><text>samizzi</text></nickname>
          <geo><uri>geo:39.59,-105.01</uri></geo>
          <org>
            <parameters><type><text>work</text></type></parameters>
            <text>Cisco</text>
          </org>
          <note>
            <text>My co-author on XEP-0292. She's cool!</text>
          </note>
          <impp>
            <parameters><type><text>work</text></type></parameters>
            <uri>xmpp:samizzi@cisco.com</uri>
          </impp>
        </vcard>
      </item>
    </publish-
    options>
    <x xmlns='jabber:x:data' _type='submit'>
      <field var='FORM_TYPE'_type='hidden'>
        <value>http://jabber.org/protocol/pubsub#publish-options</value>
      </field>
    </x>
  </pubsub>
</iq>
```

When a contact's vCard is stored in a private node, it is pushed out to all of the user’s resources that have included in their entity capabilities (XEP-0115) data a service discovery feature of "urn:xmpp:contacts+notify" (in the following example those resources are "squire" and "roundabout").

Listing 9: Publisher resources receive event notification

```xml
<message from='stpeter@stpeter.im'
to='stpeter@stpeter.im/squire'
type='headline'
id='ka92g1b5'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:xmpp:contacts'>
      <item id='9703CC4E-CF7E-4A86-9E61-2C670235F9CB'>
        <vcard xmlns="urn:ietf:params:xml:ns:vcard-4.0">
          <fn>
            <text>Samantha Mizzi</text>
          </fn>
          <n>
            <surname>Mizzi</surname>
            <given>Samantha</given>
            <additional/>
          </n>
          <nickname>
            <text>Sam</text>
          </nickname>
          <nickname>Samizzi</nickname>
          <geo>
            <uri>geo:39.59,-105.01</uri>
          </geo>
          <org>
            <parameters type="work">
              <text>Cisco</text>
            </parameters>
            <note>
              My co-author on XEP-0292. She's cool!
            </note>
          </org>
          <impp>
            <parameters type="work">
              <uri>xmpp:samizzi@cisco.com</uri>
            </parameters>
          </impp>
        </vcard>
      </item>
    </items>
  </event>
</message>
```
6 vCards of Automated Entities

Traditionally, vCards have been used on the XMPP network for entities other than human users, e.g. by XMPP servers and chatrooms. When such automated entities use vCards, it is RECOMMENDED to specify a value of "application" for the vCard4 KIND property RFC 6473. 

7 Determining Support

If an XMPP client or server supports the vCard4 namespace, it MUST advertise that fact in its responses to Service Discovery (XEP-0030)\(^{14}\) information ("disco#info") requests by returning a feature of "urn:ietf:params:xml:ns:vcard-4.0":

---

In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in Entity Capabilities (XEP-0115)\(^{15}\). However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

### 8 Security Considerations

The vCard information published to one's XMPP server is world-readable; therefore, users should exercise due caution when determining what information to include (e.g., street addresses, personal telephone numbers, or email addresses).

### 9 IANA Considerations

This document does not require interaction with the Internet Assigned Numbers Authority (IANA)\(^{16}\).

### 10 XMPP Registrar Considerations

#### 10.1 Well-Known Service Discovery Nodes

The XMPP Registrar\(^{17}\) shall include 'urn:xmpp:contact' and 'urn:xmpp:vcard4' in its registry of Nodes for Service Discovery and Publish-Subscribe at <https://xmpp.org/registrar/nodes.html>.

---


\(^{16}\)The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

\(^{17}\)The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.
11 Mapping from vcard-temp to vCard4

This section provides a more detailed description of mapping vcard-temp properties to vCard4 properties.

11.1 Properties Defined in vcard-temp but not in vCard3 or vCard4

11.1.1 DESC

The vcard-temp specification defined a <DESC/> element. This element was not part of the vCard3 schema. Mapping the vcard-temp <DESC/> element to the vCard4 NOTE property is appropriate.

Listing 13: Deprecated DESC element

```xml
<DESC>
  More information about me is located on my personal website: https://stpeter.im/
</DESC>
```

Listing 14: NOTE property

```xml
<note>
  <text>
    More information about me is located on my personal website: https://stpeter.im/
  </text>
</note>
```

11.1.2 JABBERID

The vcard-temp specification defined a <JABBERID/> element:

Listing 15: Deprecated JABBERID element

```xml
<JABBERID>stpeter@jabber.org</JABBERID>
```

Although the JABBERID field was not part of the vCard3 schema and was simply hacked into vcard-temp, RFC 4770 defined an IMPP property for instant messaging and presence addresses, which was ported to vCard4. In the vCard4 XML format, the IMPP property for a JabberID would be as follows.

11.1 MAPPING FROM VCARD-TEMP TO VCARD4

Listing 16: IMPP property

```
<impp>
  <uri>xmpp:stpeter@jabber.org</uri>
</impp>
```

11.1.3 MIDDLE

The vcard-temp specification defined a `<MIDDLE/>` element as the third allowable element within the `<N/>` (“name”) element. This element was not part of the vCard3 schema, although the Dawson drafts did contain an `<other/>` element in the third position of child elements within the `<n/>` element. It is appropriate to map the vcard-temp `<MIDDLE/>` element to the vCard4 “Additional Name” part of the “N” structured property value, which in xCard is the `<additional/>` child of the `<n/>` element.

11.2 Properties Defined Incorrectly in vcard-temp

Several of the properties in vcard-temp are defined differently in vCard3. In fact, the definitions even differ from those provisionally made in the so-called “Dawson drafts” from which vcard-temp was supposedly derived (for reference, the last of these is archived at `<http://www.watersprings.org/pub/id/draft-dawson-vcard-xml-dtd-03.txt>`). The reasons for these discrepancies are unknown. However, care must be taken in correctly mapping these properties from vcard-temp to vCard4.

11.2.1 KEY

The DTD in XEP-0054 provided this definition for the KEY element:

```
<!ELEMENT KEY ( TYPE ?, CRED )>
```

However, the DTD in the final Dawson draft provided the following definition:

```
<!ELEMENT key ( extref | b64bin )>
```

The relevant RelaxNG definition in vCard4 XML is as follows:

```
property-key = element key {
  ( value-uri | value-text )
}
```

The source of the spurious `<TYPE/>` and `<CRED/>` elements is unknown. The vcard-temp `<CRED/>` element is mapped to the vCard4 value-text construction.
11.2.2 SOUND

The DTD in XEP-0054 provided this definition for the SOUND element:

```
<!ELEMENT SOUND (PHONETIC | BINVAL | EXTVAL)>
```

However, the DTD in the final Dawson draft provided the following definition:

```
<!ELEMENT sound (extref | b64bin)>
```

The source of the spurious vcard-temp `<PHONETIC/>` element is unknown. However, it does not exist in vCard4 and therefore is simply discarded when mapping. The vcard-temp `<BINVAL/>` element is mapped to the vCard4 b64bin construction and the vcard-temp `<EXTVAL/>` element is mapped to the vCard4 extref construction.

11.2.3 VERSION

As explained in XEP-0054, the `<VERSION/>` element from the final Dawson draft was not used in vcard-temp; instead, the vcard-temp protocol used a 'version' attribute (in fact the Dawson drafts were inconsistent, since the DTD defined a `<VERSION/>` element and the body of the specification used a 'version' attribute).

11.3 Properties Defined Differently in vcard-temp, vCard3, and vCard4

The following properties are defined differently in vcard-temp and vCard4. As a result, the mappings are workable but might not preserve all information that could have been contained in vcard-temp data.

11.3.1 ADR

The following address type values allowed in vCard3 were removed from vCard4:

- DOM
- INTL
- PARCEL
- POSTAL
11.3.2 AGENT

In vCard3 and vcard-temp, the AGENT property was allowed to contain the inline vCard of someone who could act as an agent for the primary owner of the referenced vCard. In vCard4, inline vCards are disallowed. Therefore only pointers to external vCard objects are now allowed, by means of a URI.

11.3.3 ORG

The ORGUNIT property was removed from vCard4, with the result that the ORGNAME property becomes the only child of ORG.

11.3.4 SORT-STRING

The SORT-STRING property from vCard3 was renamed to SORT-AS in vCard4.

11.3.5 TEL

The following telephony type values allowed in vCard3 were removed from vCard4:

- BBS
- ISDN
- MODEM
- MSG
- PCS

In addition, in vCard4 the telephone number is represented as a tel: URI, not by means of a NUMBER property.

11.4 Properties Defined Similarly in vcard-temp, vCard3, and vCard4

The following properties are defined similarly in vcard-temp, vCard3, and vCard4. The mappings are fairly straightforward (a future version of this document might provide more detailed narrative descriptions of the mappings).

- BDAY
- CATEGORIES
11 MAPPING FROM VCARD-TEMP TO VCARD4

- EMAIL
- FN
- GEO
- LOGO
- N
- NICKNAME
- NOTE
- PHOTO (mapped to a ’data:’ URI in vCard4, see RFC 2397\(^\text{19}\))
- PRODID
- REV
- ROLE
- TITLE
- TZ
- UID
- URL

11.5 Properties Defined in vCard3 but Removed from vCard4

The following properties were defined in vCard3 but were removed from vCard4:

- CLASS
- LABEL
- MAILER

There is no mapping from these properties to vCard4.

12 Migration Tools

This section contains three tools that are intended to help developers in migrating from vcard-temp to vCard4 XML:

1. An Extensible Stylesheet Language Transformation (XSLT) script for automatically translating the vcard-temp XML format into the vCard4 XML format.

2. An example of vcard-temp data that uses most of the elements defined in XEP-0054 that can be mapped to vCard4 properties (note that some of these elements were never used in practice).

3. An example of vCard4 XML data showing the transformation of the vcard-temp example using the XSLT stylesheet.

The tools are purely informational and are not a normative part of this specification.

12.1 Extensible Stylesheet Language Transformation (XSLT) from vcard-temp to vCard4

```xml
<?xml version='1.0' encoding='UTF-8'?>
<!--
Copyright (c) 1999 - 2017 XMPP Standards Foundation
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
```
ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

-->

<!-- Author: stpeter@jabber.org -->

<!-- Version: 0.0.3 -->

<!-- Last Updated: 2012-09-12 -->

<xsl:stylesheet xmlns='urn:ietf:params:xml:ns:vcard-4.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
  <xsl:output method='xml'/>
  <xsl:template match=''/>
  <vcard>
    <!-- BEGIN VARIABLES -->
    <!-- in case we need them, count instances of the vcard-temp elements per the DTD in XEP-0054 -->
    <xsl:variable name='FN.count' select='count(/vCard/FN)'/>
    <xsl:variable name='N.count' select='count(/vCard/N)'/>
    <xsl:variable name='NICKNAME.count' select='count(/vCard/NICKNAME)'/>
    <xsl:variable name='PHOTO.count' select='count(/vCard/PHOTO)'/>
    <xsl:variable name='BDAY.count' select='count(/vCard/BDAY)'/>
    <xsl:variable name='ADR.count' select='count(/vCard/ADR)'/>
    <xsl:variable name='LABEL.count' select='count(/vCard/LABEL)'/>
    <xsl:variable name='TEL.count' select='count(/vCard/TEL)'/>
    <xsl:variable name='EMAIL.count' select='count(/vCard/EMAIL)'/>
    <xsl:variable name='JABBERID.count' select='count(/vCard/JABBERID)'/>
    <xsl:variable name='MAILER.count' select='count(/vCard/MAILER)'/>
    <xsl:variable name='TZ.count' select='count(/vCard/TZ)'/>
    <xsl:variable name='GEO.count' select='count(/vCard/GEO)'/>
    <xsl:variable name='TITLE.count' select='count(/vCard/TITLE)'/>
    <xsl:variable name='ROLE.count' select='count(/vCard/ROLE)'/>
    <xsl:variable name='LOGO.count' select='count(/vCard/LOGO)'/>
    <xsl:variable name='AGENT.count' select='count(/vCard/AGENT)'/>
    <xsl:variable name='ORG.count' select='count(/vCard/ORG)'/>
  </vcard>
</xsl:template>
</xsl:stylesheet>
<xsl:variable name='CATEGORIES.count' select='count(/vCard/CATEGORIES)'/>
<xsl:variable name='NOTE.count' select='count(/vCard/NOTE)'/>
<xsl:variable name='PRODID.count' select='count(/vCard/PRODID)' />
<xsl:variable name='REV.count' select='count(/vCard/REV)' />
<xsl:variable name='SORT-STRING.count' select='count(/vCard/SORT-STRING)' />
<xsl:variable name='SOUND.count' select='count(/vCard/SOUND)'/>
<xsl:variable name='UID.count' select='count(/vCard/UID)' />
<xsl:variable name='URL.count' select='count(/vCard/URL)' />
<xsl:variable name='CLASS.count' select='count(/vCard/CLASS)' />
<xsl:variable name='KEY.count' select='count(/vCard/KEY)' />
<xsl:variable name='DESC.count' select='count(/vCard/DESC)' />
</xsl:variable>
<!-- END VARIABLES -->
<!-- BEGIN VCARD4 OUTPUT -->
<!-- FN is required -->
<fn><text><xsl:value-of select=''/vCard/FN'/></text></fn>
<!-- N is required -->
<xsl:apply-templates select=''/vCard/N'/>
<!-- NICKNAME can be included one or more times -->
<xsl:for-each select=''/vCard/NICKNAME'>
 <nickname><text><xsl:value-of select='.'/><!--</text><!--</nickname>
</xsl:for-each>
<!-- PHOTO can be included one or more times -->
<!-- content can be either a pointer to a URL or inline binary, which is mapped to a data: URI in vCard4 -->
<xsl:for-each select=''/vCard/PHOTO'>
 <xsl:variable name='PHOTO.ext' select='count(EXTVAL)' />
 <xsl:variable name='PHOTO.type' select='TYPE'/>
 <xsl:choose>
 <xsl:when test='$PHOTO.ext=1'>
 <photo><uri><xsl:value-of select='EXTVAL'/></uri></photo>
 </xsl:when>
 <xsl:otherwise>
 <photo>
 <uri>
 <xsl:text data:></xsl:text><xsl:value-of select='TYPE'/>
 </xsl:text>;</xsl:text><xsl:value-of select='TYPE'/>
 </photo>
 </xsl:otherwise>
</xsl:choose>
</xsl:for-each>
BINVAL />
</uri>
</photo>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
<!-- BDAY can be included once -->
<xsl:if test='$BDAY.count=1'>
  <bday><date><xsl:value-of select='/vCard/BDAY'/></date></bday>
</xsl:if>
<!-- ADR can be included one or more times -->
<xsl:apply-templates select='/vCard/ADR'/>
<!-- NOTE: vcard-temp allowed a LABEL element, but it was removed from vCard4 -->
<!-- one or more TEL elements can be included -->
<xsl:apply-templates select='/vCard/TEL'/>
<!-- one or more EMAIL elements can be included -->
<xsl:apply-templates select='/vCard/EMAIL'/>
<!-- JABBERID can be included one or more times -->
<!-- we map this to the vCard4 IMPP property -->
<xsl:for-each select='/vCard/JABBERID'>
</uri></impp>
</xsl:for-each>
<!-- NOTE: vcard-temp allowed a MAILER element, but it was removed from vCard4 -->
<!-- one or more TZ elements can be included -->
<xsl:for-each select='/vCard/TZ'>
  <tz><text><xsl:value-of select='.'/></text></tz>
</xsl:for-each>
<!-- one or more GEO elements can be included -->
<xsl:for-each select='/vCard/GEO'>
</xsl:for-each>

<!-- one or more TITLE elements can be included -->
<xsl:for-each select='/vCard/TITLE'>
  <title><xsl:value-of select='.'/></title>
</xsl:for-each>

<!-- one or more ROLE elements can be included -->
<xsl:for-each select='/vCard/ROLE'>
  <role><xsl:value-of select='.'/></role>
</xsl:for-each>

<!-- one or more LOGO elements can be included -->
<!-- content is either a pointer to a URL or inline binary, which is mapped to a data: URI in vCard4 -->
<xsl:for-each select='/vCard/LOGO'>
  <xsl:variable name='LOGO.ext' select='count(EXTVAL)'/>
  <xsl:variable name='LOGO.type' select='TYPE'/>
  <xsl:choose>
    <xsl:when test='$LOGO.ext=1'>
      <logo><uri><xsl:value-of select='EXTVAL'/></uri></logo>
    </xsl:when>
    <xsl:otherwise>
      <logo>
      </logo>
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>

<!-- one or more AGENT elements can be included -->
<!-- however, inline vCards are not supported in vCard4 -->
<!-- the relevant vCard4 property is RELATED -->
<!-- only EXTVAL is supported now via URIs -->
<!-- NOTE: this element was probably unused in vcard-temp -->
<xsl:for-each select='/vCard/AGENT'>
  <xsl:variable name='AGENT.ext' select='count(EXTVAL)'/>
  <xsl:if test='$AGENT.ext=1'>
    <agent><uri><xsl:value-of select='EXTVAL'/></uri></agent>
  </xsl:if>
</xsl:for-each>
</xsl:if>
</xsl:for-each>

<!-- one or more ORG elements can be included -->
<!-- NOTE: ORGUNIT is no longer used in vcard4 -->

<xsl:for-each select='//vCard/ORG'>
<org>
  <xsl:variable name='ORGNAME.count' select='count(ORGNAME)'/>
  <xsl:if test='$ORGNAME.count=1'>
    <text><xsl:value-of select='ORGNAME'/></text>
  </xsl:if>
</org>
</xsl:for-each>

<!-- one or more CATEGORIES elements can be included -->
<!-- NOTE: this element was probably unused in vcard-temp -->

<xsl:for-each select='//vCard/CATEGORIES'>
  <categories><text><xsl:value-of select='.'/></text></categories>
</xsl:for-each>

<!-- one or more NOTE elements can be included -->
<!-- NOTE: this element was probably unused in vcard-temp -->

<xsl:for-each select='//vCard/NOTE'>
  <note><text><xsl:value-of select='.'/></text></note>
</xsl:for-each>

<!-- PRODID can be included exactly once -->
<!-- NOTE: this element was probably unused in vcard-temp -->

<xsl:if test='$PRODID.count=1'>
  <prodid><text><xsl:value-of select='.'/></text></prodid>
</xsl:if>

<!-- REV can be included exactly once -->
<!-- NOTE: this element was probably unused in vcard-temp -->

<xsl:if test='$REV.count=1'>
  <rev><timestamp><xsl:value-of select='.'/></timestamp></rev>
</xsl:if>

<!-- one or more SORT-STRING elements can be included -->
<!-- this element maps to SORT-AS in vCard4 -->

<xsl:for-each select='//vCard/SORT-STRING'>
  <sort-as><xsl:value-of select='.'/></sort-as>
</xsl:for-each>
12 MIGRATION TOOLS

<!-- one or more SOUND elements can be included -->
<!-- NOTE: for some reason, vcard-temp allowed a <PHONETIC/> child element, but that was not documented in the original Dawson I-Ds and is not supported in vCard4 -->

<xsl:for-each select='/vCard/SOUND'>
  <xsl:variable name='SOUND.ext' select='count(EXTVAL)'/>
  <xsl:variable name='SOUND.bin' select='count(BINVAL)'/>
  <xsl:choose>
    <xsl:when test='$SOUND.ext=1'>
      <sound><uri><xsl:value-of select='EXTVAL'/></uri></sound>
    </xsl:when>
    <xsl:when test='$SOUND.ext=1'>
      <sound>
        <uri><xsl:text data:audio/basic;base64>,</xsl:text><xsl:value-of select='BINVAL'/></uri>
      </sound>
    </xsl:when>
    <xsl:otherwise/>
  </xsl:choose>
</xsl:for-each>

<!-- UID can be included exactly once -->
<xsl:if test='$UID.count=1'>
  <uid><uri><xsl:value-of select='/vCard/UID'/></uri></uid>
</xsl:if>

<!-- URL can be included one or more times -->
<xsl:for-each select='/vCard/URL'>
  <url><uri><xsl:value-of select='.'/></uri></url>
</xsl:for-each>

<!-- NOTE: vcard-temp allowed a CLASS element, but it was removed from vCard4 -->

<!-- KEY can be included one or more times -->
<xsl:for-each select='/vCard/KEY'>
  <key><text><xsl:value-of select='CRED'/></text></key>
</xsl:for-each>

<!-- DESC can be included one or more times -->
<!-- this existed in vcard-temp but not vCard3 -->
<!-- mapped to the NOTE element -->
<xsl:for-each select='/vCard/DESC'>
  <note><text><xsl:value-of select='.'/></text></note>
</xsl:for-each>
</vcard>
</xsl:template>

<xsl:template match='N'>
  <n>
    <xsl:variable name='FAMILY.count' select='count(FAMILY)'/>
    <xsl:variable name='GIVEN.count' select='count(GIVEN)'/>
    <xsl:variable name='MIDDLE.count' select='count(MIDDLE)'/>
    <xsl:variable name='PREFIX.count' select='count(PREFIX)'/>
    <xsl:variable name='SUFFIX.count' select='count(SUFFIX)'/>
    <xsl:if test='$FAMILY.count=1'>
      <surname><xsl:value-of select='FAMILY'/></surname>
    </xsl:if>
    <xsl:if test='$GIVEN.count=1'>
      <given><xsl:value-of select='GIVEN'/></given>
    </xsl:if>
    <xsl:if test='$MIDDLE.count=1'>
      <additional><xsl:value-of select='MIDDLE'/></additional>
    </xsl:if>
    <xsl:if test='$PREFIX.count=1'>
      <prefix><xsl:value-of select='PREFIX'/></prefix>
    </xsl:if>
    <xsl:if test='$SUFFIX.count=1'>
      <suffix><xsl:value-of select='SUFFIX'/></suffix>
    </xsl:if>
  </n>
</xsl:template>

<xsl:template match='ADR'>
  <adr>
    <xsl:variable name='HOME.count' select='count(HOME)'/>
    <xsl:variable name='WORK.count' select='count(WORK)'/>
    <xsl:if test='count(HOME)+count(WORK)=0'>
      <!-- NOTE: vCard-temp allowed address types of POSTAL, PARCEL, DOM, and INTL, but they were removed from vCard4 -->
      <xsl:variable name='PREF.count' select='count(PREF)'/>
      <xsl:variable name='POBOX.count' select='count(POBOX)'/>
      <xsl:variable name='EXTADD.count' select='count(EXTADD)'/>
      <xsl:variable name='STREET.count' select='count(STREET)'/>
      <xsl:variable name='LOCALITY.count' select='count(LOCALITY)'/>
      <xsl:variable name='REGION.count' select='count(REGION)'/>
    </xsl:if>
  </adr>
</xsl:template>
<xsl:variable name='PCODE.count' select='count(PCODE)'/>

<!-- NOTE: yes, vcard-temp has CTRY, not COUNTRY -->
<xsl:variable name='CTRY.count' select='count(CTRY)'/>

<!-- first we count the number of vCard TYPE parameters -->
<xsl:variable name='TYPE.count' select='$HOME.count+$WORK.count'/>

<!-- now we output all the parameters -->
<xsl:if test='$TYPE.count > 0'>
  <parameters>
    <type>
      <xsl:if test='$HOME.count=1'>
        <text>home</text>
      </xsl:if>
      <xsl:if test='$WORK.count=1'>
        <text>work</text>
      </xsl:if>
    </type>
    <xsl:if test='$PREF.count=1'>
      <pref><integer>1</integer></pref>
    </xsl:if>
  </parameters>
</xsl:if>
<xsl:if test='$POBOX.count=1'>
  <pobox><xsl:value-of select='POBOX'/></pobox>
</xsl:if>
<xsl:if test='$EXTADD.count=1'>
  <ext><xsl:value-of select='EXTADD'/></ext>
</xsl:if>
<xsl:if test='$STREET.count=1'>
  <street><xsl:value-of select='STREET'/></street>
</xsl:if>
<xsl:if test='$LOCALITY.count=1'>
  <locality><xsl:value-of select='LOCALITY'/></locality>
</xsl:if>
<xsl:if test='$REGION.count=1'>
  <region><xsl:value-of select='REGION'/></region>
</xsl:if>
<xsl:if test='$PCODE.count=1'>
  <code><xsl:value-of select='PCODE'/></code>
</xsl:if>
<xsl:if test='$CTRY.count=1'>
  <country><xsl:value-of select='CTRY'/></country>
</xsl:if>
<adr>
</adr>
</xsl:template>

<xsl:template match='TEL'>
<tel>
  <xsl:variable name='HOME.count' select='count(HOME)'/>
  <xsl:variable name='WORK.count' select='count(WORK)'/>
  <xsl:variable name='TEXT.count' select='count(TEXT)'/>
  <xsl:variable name='VOICE.count' select='count(VOICE)'/>
  <xsl:variable name='FAX.count' select='count(FAX)'/>
  <xsl:variable name='CELL.count' select='count(CELL)'/>
  <xsl:variable name='VIDEO.count' select='count(VIDEO)'/>
  <xsl:variable name='PAGER.count' select='count(PAGER)'/>
  <xsl:variable name='TEXTPHONE.count' select='count(TEXTPHONE)'/>
</tel>
</xsl:template>

<!-- NOTE: vcard-temp allowed telephony types of MSG, BBS, MODEM, ISDN, and PCS but they were removed from vCard4 -->

<xsl:variable name='PREF.count' select='count(PREF)'/>
<xsl:variable name='NUMBER.count' select='count(NUMBER)'/>

<!-- first we count the number of vCard TYPE parameters -->
<xsl:variable name='TYPE.count' select='$HOME.count + $WORK.count + $TEXT.count + $VOICE.count + $FAX.count + $CELL.count + $VIDEO.count + $PAGER.count + $TEXTPHONE.count' />

<!-- now we output all the parameters -->

<xsl:if test='$TYPE.count &gt; 0'>
  <parameters>
    <type>
      <xsl:if test='$HOME.count=1'>
        <text>home</text>
      </xsl:if>
      <xsl:if test='$WORK.count=1'>
        <text>work</text>
      </xsl:if>
      <xsl:if test='$TEXT.count=1'>
        <text>text</text>
      </xsl:if>
      <xsl:if test='$VOICE.count=1'>
        <text>voice</text>
      </xsl:if>
    </type>
  </parameters>
</xsl:if>
<xsl:if test='$FAX.count=1'>
  <text>fax</text>
</xsl:if>
<xsl:if test='$CELL.count=1'>
  <text>cell</text>
</xsl:if>
<xsl:if test='$VIDEO.count=1'>
  <text>video</text>
</xsl:if>
<xsl:if test='$PAGER.count=1'>
  <text>pager</text>
</xsl:if>
<xsl:if test='$TEXTPHONE.count=1'>
  <text>texphone</text>
</xsl:if>
</type>
<xsl:if test='$PREF.count=1'>
  <pref>
    <integer>1</integer>
  </pref>
</xsl:if>
</parameters>
</xsl:if>
<xsl:if test='$NUMBER.count=1'>
  <uri>
    <xsl:text>tel:</xsl:text>
    <xsl:value-of select='NUMBER'/>
  </uri>
</xsl:if>
</tel>
</xsl:template>
<xsl:template match='EMAIL'>
  <email>
    <xsl:variable name='HOME.count' select='count(HOME)'/>
    <xsl:variable name='WORK.count' select='count(WORK)'/>

    <!-- NOTE: vcard-temp allowed email types of INTERNET and X400, but they were never in vCard3 -->
    <xsl:variable name='PREF.count' select='count(PREF)'/>
    <xsl:variable name='USERID.count' select='count(USERID)'/>

    <!-- first we count the number of vCard TYPE parameters -->
    <xsl:variable name='TYPE.count' select='$HOME.count+$WORK.count'/>

    <!-- now we output all the parameters -->
    <xsl:if test='$TYPE.count>0'>
      <parameters>
        <type>
12.2 Example of vcard-temp Data

```
<vCard>
  <FN>Peter Saint-Andre</FN>
  <N>
    <FAMILY>Saint-Andre</FAMILY>
    <GIVEN>Peter</GIVEN>
    <MIDDLE/>
  </N>
  <NICKNAME>stpeter</NICKNAME>
  <NICKNAME>psa</NICKNAME>
  <PHOTO><EXTVAL>http://stpeter.im/images/stpeter_oscon.jpg</EXTVAL></PHOTO>
  <PHOTO><EXTVAL>http://stpeter.im/images/stpeter_hell.jpg</EXTVAL></PHOTO>
  <BDAY>1966-08-06</BDAY>
  <ADR>
    <WORK/>
    <PREF/>
    <EXTADD>Suite 600</EXTADD>
    <STREET>1899 Wynkoop Street</STREET>
    <LOCALITY>Denver</LOCALITY>
    <REGION>CO</REGION>
    <PCODE>80202</PCODE>
    <CTRY>USA</CTRY>
  </ADR>
  <ADR/>
  <HOME/>
</vCard>
```
More information about me is located on my personal website: https://stpeter.im/
12.3 Example of vCard4 XML Data

```xml
<vcard xmlns="urn:ietf:params:xml:ns:vcard-4.0">
  <fn><text>Peter Saint-Andre</text></fn>
  <n>
    <surname>Saint-Andre</surname>
    <given>Peter</given>
    <additional></additional>
  </n>
  <nickname>
    <text>stpeter</text>
  </nickname>
  <nickname>
    <text>psa</text>
  </nickname>
  <photo>
    <uri>http://stpeter.im/images/stpeter_oscon.jpg</uri>
  </photo>
  <photo>
    <uri>http://stpeter.im/images/stpeter_hell.jpg</uri>
  </photo>
  <bday><date>1966-08-06</date></bday>
  <adr>
    <parameters>
      <type><text>work</text></type>
      <pref><integer>1</integer></pref>
    </parameters>
    <ext>Suite 600</ext>
    <street>1899 Wynkoop Street</street>
    <locality>Denver</locality>
    <region>CO</region>
    <code>80202</code>
    <country>USA</country>
  </adr>
  <adr>
    <parameters>
      <type><text>home</text></type>
    </parameters>
    <ext></ext>
    <street></street>
    <locality>Parker</locality>
    <region>CO</region>
    <code>80138</code>
    <country>USA</country>
  </adr>
  <tel>
    <parameters>
    </parameters>
</vcard>
```
13 ACKNOWLEDGEMENTS

Thanks to Dave Cridland, Todd Herman, Joe Hildebrand, Waqas Hussain, and Matt Miller for their feedback.