XEP-0153: vCard-Based Avatars

Peter Saint-Andre
mailto:stpeter@jabber.org
xmpp:stpeter@jabber.org
https://stpeter.im/

2006-08-16
Version 1.0

Status	Type	Short Name
Active	Historical	vcard-avatar

This document provides historical documentation of a vCard-based protocol for exchanging user avatars.

Legal

Copyright

Permissions

Warranty

Liability

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 <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1 Introduction

There exist several proposed protocols for communicating user avatar information over Jabber/XMPP (see IQ-Based Avatars [1] and User Avatar [2]). This document describes another such protocol that is in use today on the Jabber/XMPP network. This document is historical and does not purport to propose a standards-track protocol. However, a future protocol may improve on the approach documented herein.

2 Requirements

The protocol described herein seems to have been designed with the following requirements in mind:

- Enable a user to store an avatar image in his or her vCard.
- Provide notice of avatar changes via the <presence/> stanza.
- Enable a contact to retrieve a user's avatar image if the user is offline.
- Enable a contact to retrieve a user's avatar image without requesting it of the user's particular client, thus preserving bandwidth.

3 Use Cases

3.1 User Publishes Avatar

Before informing contacts of the user's avatar, the user's client first publishes the avatar data to the user's public vCard using the protocol defined in vcard-temp [3].

Listing 1: User's Client Publishes Avatar Data to vCard

```
<iq from='juliet@capulet.com'
    type='set'
    id='vc1'>
  <vCard xmlns='vcard-temp'>
    <BDAY>1476-06-09</BDAY>
    <ADR>
      <CTRY>Italy</CTRY>
      <LOCALITY>Verona</LOCALITY>
      <HOME/>
    </ADR>
```

[1] XEP-0008: IQ-Based Avatars <http://xmpp.org/extensions/xep-0008.html>.
[2] XEP-0084: User Avatar <http://xmpp.org/extensions/xep-0084.html>.
[3] XEP-0054: vcard-temp <http://xmpp.org/extensions/xep-0054.html>.

```
    <NICKNAME/>
    <N><GIVEN>Juliet</GIVEN><FAMILY>Capulet</FAMILY></N>
    <EMAIL>jcapulet@shakespeare.lit</EMAIL>
    <PHOTO>
      <TYPE>image/jpeg</TYPE>
      <BINVAL>
        Base64-encoded-avatar-file-here!
      </BINVAL>
    </PHOTO>
  </vCard>
</iq>
```

Listing 2: User's Server Acknowledges Publish

```
<iq to='juliet@capulet.com' type='result' id='vc1'/>
```

Next, the user's client computes the SHA1 hash of the avatar image data itself (not the base64-encoded version) in accordance with RFC 3174 [4]. This hash is then included in the user's presence information as the XML character data of the <photo/> child of an <x/> element qualified by the 'vcard-temp:x:update' namespace, as shown in the following example:

Listing 3: User's Client Includes Avatar Hash in Presence Broadcast

```
<presence from='juliet@capulet.com/balcony'>
  <x xmlns='vcard-temp:x:update'>
    <photo>sha1-hash-of-image</photo>
  </x>
</presence>
```

The user's server then broadcasts that presence information to all contacts who are subscribed to the user's presence information.

3.2 Contact Retrieves Avatar

When the recipient's client receives the hash of the avatar image, it SHOULD check the hash to determine if it already has a cached copy of that avatar image. If not, it retrieves the sender's full vCard in accordance with the protocol flow describerd in XEP-0054 (note that this request is sent to the user's bare JID, not full JID):

Listing 4: Contact's Client Requests User's vCard

```
<iq from='romeo@montague.net/orchard'
    to='juliet@capulet.com'
    type='get'
    id='vc2'>
```

[4] RFC 3174: US Secure Hash Algorithm 1 (SHA1) <http://tools.ietf.org/html/rfc3174>.

```
  <vCard xmlns='vcard-temp'/>
</iq>
```

Listing 5: Server Returns vCard on Behalf of User

```
<iq from='juliet@capulet.com'
    to='romeo@montague.net/orchard'
    type='result'
    id='vc2'>
  <vCard xmlns='vcard-temp'>
    <BDAY>1476-06-09</BDAY>
    <ADR>
      <CTRY>Italy</CTRY>
      <LOCALITY>Verona</LOCALITY>
      <HOME/>
    </ADR>
    <NICKNAME/>
    <N><GIVEN>Juliet</GIVEN><FAMILY>Capulet</FAMILY></N>
    <EMAIL>jcapulet@shakespeare.lit</EMAIL>
    <PHOTO>
      <TYPE>image/jpeg</TYPE>
      <BINVAL>
        Base64-encoded-avatar-file-here!
      </BINVAL>
    </PHOTO>
  </vCard>
</iq>
```

4 Business Rules

4.1 Inclusion of Update Data in Presence

The following rules apply to inclusion of the update child element (<x xmlns='vcard-temp:x:update'/>) in presence broadcasts:

1. If a client supports the protocol defined herein, it MUST include the update child element in every presence broadcast it sends and SHOULD also include the update child in directed presence stanzas (e.g., directed presence sent when joining Multi-User Chat [5] rooms).

2. If a client is not yet ready to advertise an image, it MUST send an empty update child element, i.e.:

[5] XEP-0045: Multi-User Chat <http://xmpp.org/extensions/xep-0045.html>.

Listing 6: User Is Not Ready to Advertise an Image

```
<presence>
  <x xmlns='vcard-temp:x:update'/>
</presence>
```

3. If there is no avatar image to be advertised, the photo element MUST be empty, i.e.:

Listing 7: No Image to be Advertised

```
<presence>
  <x xmlns='vcard-temp:x:update'>
    <photo/>
  </x>
</presence>
```

If the client subsequently obtains an avatar image (e.g., by updating or retrieving the vCard), it SHOULD then publish a new <presence/> stanza with character data in the <photo/> element.
Note: This enables recipients to distinguish between the absence of an image (empty photo element) and mere support for the protocol (empty update child).

4.2 Downloading and Uploading the vCard

The following rules apply to downloading and uploading the vCard:

1. A client MUST NOT advertise an avatar image without first downloading the current vCard. Once it has done this, it MAY advertise an image. However, a client MUST advertise an image if it has just uploaded the vCard with a new avatar image. In this case, the client MAY choose not to redownload the vCard to verify its contents.

2. Within a given session, a client MUST NOT attempt to upload a given avatar image more than once. The client MAY upload the avatar image to the vCard on login and after that MUST NOT upload the vCard again unless the user actively changes the avatar image.

3. The client MUST NOT poll for new versions of the user's vCard in order to determine whether to update the avatar image hash.

4.3 Multiple Resources

Jabber/XMPP allows multiple resources to authenticate for the same JID simultaneously. This introduces the potential of conflict between the resources regarding the user's avatar image.

The following rules apply when a client receives a presence broadcast from another resource of its own JID:

1. If the presence stanza received from the other resource does not contain the update child element, then the other resource does not support vCard-based avatars. That resource could modify the contents of the vCard (including the photo element); because polling for vCard updates is not allowed, the client MUST stop advertising the avatar image hash. However, the client MAY reset its hash if all instances of non-conforming resources have gone offline.

2. If the presence stanza received from the other resource contains the update child element, then the other resource conforms to the protocol for vCard-based avatars. There are three possible scenarios:

   - If the update child element is empty, then the other resource supports the protocol but does not have its own avatar image. Therefore the client can ignore the other resource and continue to broadcast the existing image hash.

   - If the update child element contains an empty photo element, then the other resource has updated the vCard with an empty BINVAL. Therefore the client MUST retrieve the vCard. If the retrieved vCard contains a photo element with an empty BINVAL, then the client MUST stop advertising the old image.

   - If the update child element contains a non-empty photo element, then the client MUST compare the image hashes. If the hashes are identical, then the client can ignore the other resource and continue to broadcast the existing image hash. If the hashes are different, then the client MUST NOT attempt to resolve the conflict by uploading its avatar image again. Instead, it MUST defer to the content of the retrieved vCard by resetting its image hash (see below) and providing that hash in future presence broadcasts.

4.4 Resetting the Image Hash

Resetting the image hash consists of the following steps:

1. Immediately send out a presence element with an empty update child element (containing no photo element).

2. Download the vCard from the server.

3. If the BINVAL is empty or missing, advertise an empty photo element in future presence broadcasts.

4. If the BINVAL contains image data, calculate the hash of image and advertise that hash in future presence broadcasts.

4.5 XML Syntax

The following rules apply to the XML syntax:

1. The <PHOTO/> element SHOULD contain a <BINVAL/> child whose XML character data is Base64-encoded data for the avatar image.

2. The <PHOTO/> element SHOULD NOT contain an <EXTVAL/> that points to a URI for the image file.

3. The <PHOTO/> element MUST NOT contain the avatar image itself.

4. The <PHOTO/> element SHOULD contain a <TYPE/> child whose XML character data specifies the content-type of the image data. The XML character data SHOULD be ”image/gif”, ”image/jpeg”, or ”image/png”.

5. The <PHOTO/> element MUST NOT possess a ’mime-type’ attribute.

4.6 Image Restrictions

The following rules apply to images:

1. The image SHOULD use less than eight kilobytes (8k) of data; this restriction is to be enforced by the publishing client.

2. The image height and width SHOULD be between thirty-two (32) and ninety-six (96) pixels; the recommended size is sixty-four (64) pixels high and sixty-four (64) pixels wide.

3. The image SHOULD be square.

4. The image content type [6] SHOULD be image/gif, image/jpeg, or image/png; support for the "image/png" content type is REQUIRED, support for the "image/gif" and "image/jpeg" content types is RECOMMENDED, and support for any other content type is OPTIONAL.

5. The image data MUST conform to the base64Binary datatype [7] and thus be encoded in accordance with Section 6.8 of RFC 2045 [8], which recommends that base64 data should have lines limited to at most 76 characters in length. However, any whitespace characters (e.g., '\r' and '\n') MUST be ignored.

5 Implementation Notes

The XML character data of the <TYPE/> element is a hint. If the XML character data of the <TYPE/> specifies a content type that does not match the data provided in the <BINVAL/> element, the processing application MUST adhere to the content type of the actual image data and MUST ignore the <TYPE/>. If the <TYPE/> is something other than image/gif, image/jpeg, or image/png, it SHOULD be ignored.
If the image data exceeds the 8 KB restriction, the processing application SHOULD process the data.

6 Security Considerations

This document introduces no security considerations above and beyond those described in XMPP Core [9], XMPP IM [10], and vcard-temp [11].

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [12].

[6] The IANA registry of content types is located at <http://www.iana.org/assignments/media-types/>.
[7] See <http://www.w3.org/TR/xmlschema-2/#base64Binary>.
[8] RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies <http://tools.ietf.org/html/rfc2045>.
[9] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[10] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[11] XEP-0054: vcard-temp <http://xmpp.org/extensions/xep-0054.html>.
[12] 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/>.

8 XMPP Registrar Considerations

8.1 Protocol Namespaces

The XMPP Registrar [13] includes 'vcard-temp:x:update' in its registry of protocol namespaces (see <http://xmpp.org/registrar/namespaces.html>).

9 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='vcard-temp:x:update'
    xmlns='vcard-temp:x:update'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0153: http://www.xmpp.org/extensions/xep-0153.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='x'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='photo' minOccurs='0' type='xs:base64Binary'/
            >
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

10 Acknowledgements

The author wishes to thank the helpful developers who have implemented this protocol and provided feedback regarding its documentation.

[13] 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 <http://xmpp.org/registrar/>.