XEP-0008: IQ-Based Avatars

Thomas Muldowney
mailto:temas@jabber.org
xmpp:temas@jabber.org

Julian Missig
mailto:julian@jabber.org
xmpp:julian@jabber.org

Jens Alfke
mailto:jens@mac.com

Peter Millard

2005-06-16
Version 0.3

Status	Type	Short Name
Deferred	Historical	None

This specification provides historical documentation of an IQ-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

Many communication applications now allow for the association of a small image or buddy icon (avatar) with a user of that application. The avatar is not intended to be a defining portrait of the user, but rather a simple expression of the user's appearance, mood, status, and the like. This proposal outlines a way to incorporate avatars into the current Jabber platform.

2 Image Requirements

Certain restrictions are placed upon the image. First, the image height and width must be between thirty-two (32) and sixty-four (64) pixels. The suggested size is sixty-four (64) pixels high and sixty-four (64) pixels wide [1]. Images should be square, but this is not required. Images should be in GIF, JPEG, or PNG format, although it is possible that in future revisions of this spec more formats will be allowed. Finally, images must use less than eight (8) kilobytes of data.

3 Avatar Availability

There are two methods of showing that a client has an avatar available:

1. Embedding the jabber:x:avatar namespace within <presence/> packets using Jabber's <x/> element
2. Displaying the jabber:iq:avatar namespace in browse requests

Partly because Jabber browsing is relatively undeveloped, this proposal focuses on the first option.
The <x/> element in the jabber:x:avatar namespace contains a SHA1 hash (hexadecimal, not base64) of the image data itself (not the base64-encoded version) in a <hash/> element. (Because the odds of creating an identical hash are small, the hash is considered unique to the image and can be used to cache images between client sessions, resulting in fewer requests for the image.) The initial announcement of the availability of an avatar is done by sending a presence packet with the jabber:x:avatar information, as follows:

```
<presence>
  <x xmlns='jabber:x:avatar'>
    <hash>SHA1 of image data</hash>
  </x>
</presence>
```

[1] It is highly recommended that clients never scale up images when displaying them.

If the avatar-generating user changes the avatar, a new presence packet is sent out with the updated information:

```
<presence>
  <x xmlns='jabber:x:avatar'>
    <hash>SHA1 of new image data</hash>
  </x>
</presence>
```

To disable the avatar, the avatar-generating user's client will send a presence packet with the jabber:x:avatar namespace but with no hash information:

```
<presence>
  <x xmlns='jabber:x:avatar'/>
</presence>
```

Clients should send the current avatar hash in every <presence/> packet even if the avatar has not changed. Remember that other clients logging in will receive a copy of the most recent <presence/> element, which should therefore always contain the most recent avatar hash. However, if the client's connection is lost unexpectedly or the client disconnects without sending an unavailable presence, the server will send other clients a <presence/> element containing no jabber:x:avatar extension. Therefore if, after receiving one or more presence packets containing jabber:x:avatar information, an avatar-receiving client receives a presence packet that does not include the jabber:x:avatar namespace, it is recommended that the client save the avatar image until the next set of information is received. In this case the avatar-generating client might send something as simple as the following:

```
<presence/>
```

4 Avatar Retrieval

There are two methods for retrieving the actual avatar data:

1. An exchange between clients of <iq/> elements in the jabber:iq:avatar namespace
2. Public XML storage from the avatar-generating client to the server and public XML retrieval from the server to the avatar-requesting client (see Private XML Storage [2]).

The first of these methods is preferred. On this model, a query is sent directly to the avatar-generating client using an <iq/> element of type "get" in the jabber:iq:avatar namespace [3] [4]:

[2] XEP-0049: Private XML Storage <http://xmpp.org/extensions/xep-0049.html>.

[3] Whenever possible, the avatar-requesting client should attempt to determine if the avatar-generating client has an avatar available before requesting it.

[4] It is suggested that no request be made if it is known (such as through a browse reply) that a client does not support the jabber:iq:avatar namespace.

```
<iq id='2' type='get' to='user@server/resource'>
  <query xmlns='jabber:iq:avatar'/>
</iq>
```

The avatar-generating client will reply with an <iq/> element of type "result" in the jabber:iq:avatar namespace; this reply will contain a query element that in turn contains a <data/> element with the MIME type in the 'mimetype' attribute and the data base64-encoded in the body of the <data/> element:

```
<iq id='2' type='result' to='user@server/resource'>
  <query xmlns='jabber:iq:avatar'>
    <data mimetype='image/jpeg'>
      Base64-Encoded Data
    </data>
  </query>
</iq>
```

If the first method fails, the second method that should be attempted by sending a request to the server for the avatar-generating user's public XML containing the avatar data. This data is to be stored in the storage:client:avatar namespace. This method presumes that the avatar-generating client has already stored its avatar data on the server:

```
<iq id='0' type='set' to='user@server'>
  <query xmlns='storage:client:avatar'>
    <data mimetype='image/jpeg'>
      Base64 Encoded Data
    </data>
  </query>
</iq>
```

Once such data has been set, the avatar can be retrieved by any requesting client from the avatar-generating client's public XML storage:

```
<iq id='1' type='get' to='user@server'>
  <query xmlns='storage:client:avatar'/>
</iq>
```

5 Future Considerations

It is acknowledged that sending avatar information within presence packets is less than desirable in many respects (e.g., in network traffic generated); however, there currently exists in Jabber no generic framework for addressing these shortcomings. Possible solutions on the horizon include live browsing and a pub/sub model, but these are still embryonic and

experimental. Once something of that nature is accepted by the Council, the avatar spec will be modified to work within that framework rather than by attaching to presence.

6 Author Note

Peter Millard, a co-author of this specification from version 0.1 through version 0.3, died on April 26, 2006.