XEP-0084: User Avatar

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

Peter Millard

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

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

2016-07-09
Version 1.1.1

Status	Type	Short Name
Draft	Standards Track	avatar

This document defines an XMPP protocol extension for exchanging user avatars, which are small images or icons associated with human users. The protocol specifies payload formats for both avatar metadata and the image data itself. The payload formats are typically transported using the personal eventing profile of XMPP publish-subscribe as specified in XEP-0163.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Many communication applications allow for the association of a small image or icon with a user of that application. Usually, such an ”avatar” is not intended to be an accurate picture of the user’s actual physical appearance, but rather a representation (often fanciful) of the user’s desired self-image or a transient state of the user (such as a mood or activity). This document defines a way to incorporate avatars into current Jabber/XMPP systems by layering this functionality on top of the XMPP Publish-Subscribe (XEP-0060) [1] extension (”pubsub”), specifically the Personal Eventing Protocol (XEP-0163) [2] subset (”PEP”), which is designed for use in the context of XMPP instant messaging and presence systems that conform to RFC 3921 [3].

The protocol defined herein uses two pubsub nodes: one node for ”metadata” about the avatar state (called the ”metadata node”) and one for the avatar data itself (called the ”data node”). This separation of metadata from data conserves bandwidth and enables both the publisher and the subscriber to cache the avatar data. (For example, a user might toggle between two or three avatars, in which case the user’s contacts can display a locally cached version of the images without having to retrieve or receive the full image each time.)

This protocol also allows storage of avatar data at a URL accessible via HTTP (see RFC 2616 [4]). [5] This can be helpful as a fallback mechanism if a pubsub-aware data repository is not available. It also makes it possible for avatar images to be hosted on public websites (e.g., an end-user-oriented community site) and retrieved from that site rather than handled directly by the publishing client in any fashion.

Finally, this protocol also enables XMPP applications to optionally integrate with third-party services that host user avatars (e.g., online gaming systems and virtual worlds).

It is intended that this specification will supersede both IQ-Based Avatars (XEP-0008) [6] and vCard-Based Avatars (XEP-0153) [7] once the PEP subset of XMPP publish-subscribe is implemented and deployed widely enough.

2 Requirements

This document addresses the following use cases for avatar publishers:

1. Publishing avatar data
2. Updating metadata about the current avatar

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[3] RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc3921>.
[4] RFC 2616: Hypertext Transport Protocol -- HTTP/1.1 <http://tools.ietf.org/html/rfc2616>.
[5] By ”accessible via HTTP” is meant that the data is available at an http: or https: URI.
[6] XEP-0008: IQ-Based Avatars <https://xmpp.org/extensions/xep-0008.html>.
[7] XEP-0153: vCard-Based Avatars <https://xmpp.org/extensions/xep-0153.html>.

3. Disabling avatars

This document addresses the following use cases for avatar subscribers:

1. Discovering avatar availability
2. Receiving notification of avatar changes
3. Retrieving avatar data via pubsub
4. Retrieving avatar data via HTTP

3 Basic Process Flow

The process for publishing and updating user avatars is as follows:

1. User publishes avatar data for ”image/png” content-type to data node and optionally publishes other content-types to HTTP URLs.
2. User publishes notification of updated avatar to metadata node, with ItemID that matches SHA-1 hash of image data for ”image/png” content-type (note: this is a hash of the image data itself, not the base64-encoded version).
3. Subscribers receive notification.
4. Optionally (and if necessary), subscribers retrieve avatar data identified by ItemID from data node using pubsub retrieve-items feature (or via HTTP).
5. Optionally, user disables avatar display.

This process flow is described more fully in the following sections.
Note: Before publishing avatar data and metadata, the user MUST determine if his or her server supports the PEP subset of pubsub by following the procedures specified in XEP-0163, since such support simplifies avatar publication. The following examples assume the availability of a PEP service.

3.1 User Publishes Data

Before updating the avatar metadata node, the publisher MUST make sure that the avatar data is available at the data node or URL. When publishing the avatar data to the data node, the publisher MUST ensure that the value of the pubsub ItemID is a SHA-1 hash of the data for the ”image/png” content-type (this is used by the subscriber to determine if a locally cached copy can be displayed).
The following example illustrates the XML structure to be sent when publishing avatar data to the data node.

Listing 1: Publishing avatar data to data node

```
<iq type='set' from='juliet@capulet.lit/chamber' id='publish1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:avatar:data'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <data xmlns='urn:xmpp:avatar:data'>
          qANQR1DBwU4DX7jmYZnncm...
        </data>
      </item>
    </publish>
  </pubsub>
</iq>
```

Listing 2: Pubsub service replies with success

```
<iq type='result' to='juliet@capulet.lit/chamber' id='publish1'/>
```

If the avatar will be made available via HTTP instead of a pubsub data node, the publisher MUST either verify that the avatar exists at the HTTP URL or publish it via standard HTTP methods (such methods are out of scope for this specification; refer to RFC 2616).

3.2 User Publishes Metadata Notification

Whenever the publisher wishes to change its current avatar, it MUST update the metadata node. As with the data node, the publisher MUST ensure that the value of the pubsub ItemID is a SHA-1 hash of the data for the "image/png" content-type (the match between the ItemID of the data node and metadata node is used by the subscriber to determine if a locally cached copy can be displayed).
The following example shows metadata specifying avatar data that is available in only one format ("image/png") and accessible only at the data node.

Listing 3: Publishing avatar metadata

```
<iq type='set' from='juliet@capulet.lit/chamber' id='publish2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:avatar:metadata'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <metadata xmlns='urn:xmpp:avatar:metadata'>
          <info bytes='12345'
                id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'
                height='64'
                type='image/png'
                width='64'/>
        </metadata>
      </item>
    </publish>
  </pubsub>
```

```
</iq>
```

The following example shows metadata specifying avatar data that is available at an HTTP URL.

Listing 4: Publishing avatar metadata

```
<iq type='set' from='juliet@capulet.lit/chamber' id='publish2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:avatar:metadata'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <metadata xmlns='urn:xmpp:avatar:metadata'>
          <info bytes='23456'
                height='64'
                id='357a8123a30844a3aa99861b6349264ba67a5694'
                type='image/gif'
                url='http://avatars.example.org/happy.gif'
                width='64'/>
        </metadata>
      </item>
    </publish>
  </pubsub>
</iq>
```

3.3 Subscribers Receive Metadata Notification

The user's virtual pubsub service would then send the metadata notification to entities that have subscribed to the user's metadata node or contacts who have advertised an interest in receiving avatar metadata by including a Entity Capabilities (XEP-0115) [8] feature of "urn:xmpp:avatar:metadata+notify".

Listing 5: Subscribers receive avatar metadata notification

```
<message to='romeo@montague.lit/home' from='juliet@capulet.lit'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:xmpp:avatar:metadata'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <metadata xmlns='urn:xmpp:avatar:metadata'>
          <info bytes='12345'
                height='64'
                id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'
                type='image/png'
                width='64'/>
        </metadata>
      </item>
    </items>
  </event>
```

[8] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

```
  <addresses xmlns='http://jabber.org/protocol/address'>
    <address type='replyto' jid='juliet@capulet.lit/chamber'/>
  </addresses>
</message>
```

As shown, depending on node configuration, the item may include Extended Stanza Addressing (XEP-0033) [9] information about the publishing resource (see XEP-0060 for details).

3.4 Subscribers Retrieve Data

Upon receiving the notification, each subscriber SHOULD determine if it has a locally cached copy of that avatar (which it can do by searching for an image identified by the ItemID). If the subscriber already has a locally cached copy of the avatar image, it MUST NOT retrieve the image data.
If the subscriber does not have a locally cached copy of the avatar image, it SHOULD retrieve the data. It can do this by sending a pubsub retrieve-items request to the data node, specifying the appropriate ItemID.

Listing 6: Subscriber requests last item by ItemID

```
<iq type='get'
    from='romeo@montague.lit/home'
    to='juliet@capulet.lit'
    id='retrieve1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:avatar:data'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'/>
    </items>
  </pubsub>
</iq>
```

The PEP service running at the user's server then SHOULD return the avatar data.

Listing 7: PEP service returns avatar data

```
<iq type='result'
    from='juliet@capulet.lit'
    to='romeo@montague.lit/home'
    id='retrieve1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:avatar:data'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <data xmlns='urn:xmpp:avatar:data'>
          qANQR1DBwU4DX7jmYZnncm...
        </data>
```

[9] XEP-0033: Extended Stanza Addressing <https://xmpp.org/extensions/xep-0033.html>.

```
      </item>
    </items>
  </pubsub>
</iq>
```

If the <info/> element sent to the metadata node possesses a 'url' attribute, the avatar data is hosted at a URL. Therefore, in order to retrieve the avatar image data for that content-type, the requesting entity MUST send an HTTP request to the specified URL. Methods for doing so are out of scope for this specification (see RFC 2616).

3.5 Publisher Disables Avatar Publishing

In order to temporarily disable avatar publishing, the user publishes an empty <metadata/> element to the metadata node.

Listing 8: Temporarily disabling avatar publishing

```
<iq type='set' from='juliet@capulet.lit/chamber' id='publish3'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:avatar:metadata'>
      <item>
        <metadata xmlns='urn:xmpp:avatar:metadata'/>
      </item>
    </publish>
  </pubsub>
</iq>
```

As before, subscribers to the metadata node would then receive the notification.

Listing 9: Subscribers receive avatar metadata notification

```
<message to='romeo@montague.lit/home' from='juliet@capulet.lit'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:xmpp:avatar:metadata'>
      <item>
        <metadata xmlns='urn:xmpp:avatar:metadata'/>
      </item>
    </items>
  </event>
</message>
```

Note: In an earlier version of this specification, the user indicated that it wanted to disable publishing by sending a <metadata/> element containing a <stop/> child element. For consistency with other PEP payload formats, support for the <stop/> element is *deprecated*.

4 Protocol Syntax

The PEP subset of pubsub requires that there shall exist a one-to-one relationship between namespaces and nodes. Because the protocol defined herein stipulates the use of two nodes (one for avatar data and one for avatar metadata), we define two namespaces, each with a corresponding root element:

- <data xmlns='urn:xmpp:avatar:data'/>
- <metadata xmlns='urn:xmpp:avatar:metadata'/>

These are further specified below.

4.1 Data Element

The <data/> element is used to communicate the avatar data itself, and only for the "image/png" content-type (support for which is REQUIRED).

```
<data xmlns='urn:xmpp:avatar:data'>
  IMAGE DATA
</data>
```

The XML character data MUST represent the image data for the avatar with a content-type of "image/png", Base64-encoded in accordance with Section 4 of RFC 4648 [10]. (Note: Line feeds SHOULD NOT be added but MUST be accepted.)
The <data/> element MUST NOT possess any attributes.
Support for the <data/> element is REQUIRED.

4.2 Metadata Element

The <metadata/> element is used to communicate information about the avatar. There are two allowable children of the <metadata/> element:

- <info/>
- <pointer/>

These are further specified below.
In addition, the <metadata/> element MAY be empty (i.e., contain no child elements); this form is used to disable avatar publishing.

[10] RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.

4.2.1 Info Element

The <info/> child element is used to communicate avatar metadata. Support for the <info/> element is REQUIRED.

```
<metadata xmlns='urn:xmpp:avatar:metadata'>
  <info bytes='size-of-image-data-in-bytes'
        height='image-height-in-pixels'
        id='SHA-1-hash-of-image-data'
        type='content-type-of-image-data'
        url='HTTP-URL-for-image-data'
        width='image-width-in-pixels'/>
</metadata>
```

The <info/> child element MUST be empty.
The defined attributes of the <info/> element are specified in the following table.

Name	Definition	Inclusion
bytes	The size of the image data in bytes.	REQUIRED
height	The height of the image in pixels.	RECOMMENDED
id	A hash of the image data for the specified content-type, where the hash is produced in accordance with the SHA-1 algorithm as specified in RFC 3174 RFC 3174: US Secure Hash Algorithm 1 (SHA1) <http://tools.ietf.org/html/rfc3174>. (with binary output).	REQUIRED
type	The IANA-registered content type of the image data.	REQUIRED
url	The http: or https: URL at which the image data file is hosted; this attribute MUST NOT be included unless the image data file can be retrieved via HTTP.	OPTIONAL
width	The width of the image in pixels	RECOMMENDED

The <metadata/> root element MAY contain more than one <info/> element. Each <info/> element MUST specify metadata for the same avatar image but in alternate content-types (e.g., "image/png", "image/gif", and "image/jpeg"), and one of the formats MUST be "image/png" to ensure interoperability. The value of the 'type' attribute MUST be an IANA-registered content type of type "image" or "video". [11] Support for the "image/png" content type is

[11] The IANA registry of content types is located at <http://www.iana.org/assignments/media-types/>.

REQUIRED. Support for the "image/gif" and "image/jpeg" content types is RECOMMENDED. Support for any other content type is OPTIONAL.

4.2.2 Pointer Element

The <pointer/> child element is used to point to an avatar that is not published via pubsub or HTTP, but rather is provided by a third-party service such as an online gaming system or virtual world.

```
<metadata xmlns='urn:xmpp:avatar:metadata'>
  <pointer>
    ... APPLICATION-SPECIFIC DATA ...
  </pointer>
</metadata>
```

The <pointer/> element MAY possess the following attributes if the publishing application has the relevant information:

- *bytes* -- The size of the image data in bytes.
- *height* -- The height of the image in pixels.
- *id* -- The SHA-1 hash of the image data for the specified content-type.
- *type* -- The IANA-registered content type of the image data.
- *width* -- The width of the image in pixels.

The content of the <pointer/> element MUST be a properly-namespaced child element that specifies information about how to retrieve the avatar from the third-party service. The structure for any such child element is out of scope for this document.
Even if the <pointer> element is included, it MUST be preceded by at least one instance of the <info/> element so that implementations that do not support the <pointer/> element can display a "fallback" format of the avatar (at a minimum, "image/png").
Support for the <pointer/> element is OPTIONAL.

5 Additional Examples

5.1 Metadata With Multiple Content-Types

The following example shows metadata specifying avatar data that is available in multiple formats ("image/png", "image/gif", and "image/mng"), where the "image/png" content-type is available only at the data node and the other content-types are available HTTP URLs.

Listing 10: Publishing avatar metadata (multiple formats)

```
<iq type='set' from='juliet@capulet.lit/chamber' id='publish3'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:avatar:metadata'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <metadata xmlns='urn:xmpp:avatar:metadata'>
          <info bytes='12345'
                height='64'
                id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'
                type='image/png'
                width='64'/>
          <info bytes='12345'
                height='64'
                id='e279f80c38f99c1e7e53e262b440993b2f7eea57'
                type='image/png'
                url='http://avatars.example.org/happy.png'
                width='64'/>
          <info bytes='23456'
                height='64'
                id='357a8123a30844a3aa99861b6349264ba67a5694'
                type='image/gif'
                url='http://avatars.example.org/happy.gif'
                width='64'/>
          <info bytes='78912'
                height='64'
                id='03a179fe37bd5d6bf9c2e1e592a14ae7814e31da'
                type='image/mng'
                url='http://avatars.example.org/happy.mng'
                width='64'/>
        </metadata>
      </item>
    </publish>
  </pubsub>
</iq>
```

In the foregoing example, the image encapsulated in the "image/png" content type is available both at a pubsub data node and at an HTTP URL; therefore it is included twice (the second time with a 'url' attribute).

5.2 Metadata With Pointer

The following example shows metadata specifying avatar data that is available in "image/png" at the data node and also with a pointer to an external service.

Listing 11: Publishing avatar metadata (with pointer)

```
<iq type='set' from='juliet@capulet.lit/chamber' id='publish4'>
```

```
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:avatar:metadata'>
      <item id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'>
        <metadata xmlns='urn:xmpp:avatar:metadata'>
          <info bytes='12345'
                height='64'
                id='111f4b3c50d7b0df729d299bc6f8e9ef9066971f'
                type='image/png'
                width='64'/>
          <pointer>
            <x xmlns='http://example.com/virtualworlds'>
              <game>Ancapistan</game>
              <character>Kropotkin</character>
            </x>
          </pointer>
        </metadata>
      </item>
    </publish>
  </pubsub>
</iq>
```

6 Service Discovery

6.1 Discovering Avatar Availability

The pubsub "auto-subscribe" and "filtered-notifications" features enable a contact to automatically subscribe to a user's avatar. However, a contact can also explicitly determine if another user publishes avatars using this protocol by sending a Service Discovery (XEP-0030) [12] items ("disco#items") request to the user's bare JID <localpart@domain.tld>.

Listing 12: Disco items request

```
<iq type='get'
    from='romeo@montague.lit/orchard'
    to='juliet@capulet.lit'
    id='items1'>
  <query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

If the user publishes avatar data to an PEP node, the result MUST contain the appropriate items.

Listing 13: Disco items result

```
<iq type='result'
    from='juliet@capulet.lit'
```

[12] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
    to='romeo@montague.lit/orchard'
    id='items1'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <item jid='juliet@capulet.lit' node='urn:xmpp:avatar:data'/>
    <item jid='juliet@capulet.lit' node='urn:xmpp:avatar:metadata'/>
  </query>
</iq>
```

The contact then MAY subscribe to the metadata node following the protocol specified in XEP-0060. However, the contact SHOULD NOT subscribe to the data node (instead, it SHOULD simply retrieve items from that node when needed, as described above).

7 Implementation Notes

7.1 Multiple Resources

If a user has multiple online resources at the same time, each resource MAY publish a different avatar. The PEP service SHOULD include the "replyto" address of the publishing resource as shown above in order to facilitate differentiation between per-resource avatars.

7.2 Avatar Synchronization

When a user logs in with a new resource and before publishing an avatar, its client SHOULD retrieve its last published avatar, either automatically by sending presence with the appropriate entity capabilities information (see XEP-0115) or using the "retrieve-items" method described in XEP-0060.

7.3 Image Handling

It is the responsibility of the receiving application to determine which avatar format to retrieve (e.g., "image/gif" rather than "image/png") and to determine the appropriate method for retrieval (e.g., HTTP rather than pubsub).
The receiving application SHOULD NOT scale up an image when displaying it.
If an avatar is not available for a contact, the receiving application MAY display the contact's photo, e.g., as provided in the contact's vCard (see vcard-temp (XEP-0054) [13]) or other profile information.

[13] XEP-0054: vcard-temp <https://xmpp.org/extensions/xep-0054.html>.

8 Security Considerations

See XEP-0060 and XEP-0163 regarding security considerations related to the underlying transport protocol.
It is possible that output of the SHA-1 hashing algorithm can result in collisions; however, the use of SHA-1 in producing a hash of the avatar data is not security-critical.

9 IANA Considerations

This document makes use of IANA-registered content types, but requires no interaction with the Internet Assigned Numbers Authority (IANA) [14].

10 XMPP Registrar Considerations

10.1 Protocol Namespaces

The XMPP Registrar [15] includes "urn:xmpp:avatar:data" and "urn:xmpp:avatar:metadata" in its registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).

11 XML Schema

11.1 Data Namespace

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:avatar:data'
    xmlns='urn:xmpp:avatar:data'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0084: http://www.xmpp.org/extensions/xep-0084.html
    </xs:documentation>
```

[14] 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/>.

[15] 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/>.

```
  </xs:annotation>

  <xs:element name='data' type='xs:base64Binary'/>

</xs:schema>
```

11.2 Metadata Namespace

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:avatar:metadata'
    xmlns='urn:xmpp:avatar:metadata'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0084: http://www.xmpp.org/extensions/xep-0084.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='metadata'>
    <xs:complexType>
      <xs:choice>
        <xs:sequence minOccurs='0' maxOccurs='1'>
          <xs:element ref='info' minOccurs='1' maxOccurs='unbounded'/>
          <xs:element ref='pointer' minOccurs='0' maxOccurs='unbounded
              '/>
        </xs:sequence>
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:element name='info'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
          <xs:attribute name='bytes' type='xs:unsignedShort' use='
              required'/>
          <xs:attribute name='height' type='xs:unsignedByte' use='
              optional'/>
          <xs:attribute name='id' type='xs:string' use='required'/>
          <xs:attribute name='type' type='xs:string' use='required'/>
          <xs:attribute name='url' type='xs:anyURI' use='optional'/>
          <xs:attribute name='width' type='xs:unsignedByte' use='
              optional'/>
```

```
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='pointer'>
    <xs:complexType>
      <xs:sequence>
        <xs:any namespace='##other'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

12 Author Note

Peter Millard, a co-author of this specification from version 0.1 through version 0.7, died on April 26, 2006. The remaining authors are thankful for his work on user avatars.