Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 5, 2009

Heiner Wolf
virtual-presence.org
March 2009

Virtual Presence Identity
draft-wolf-vp-identity-00

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 September 2, 2007.

Copyright Notice

Note

This protocol is proposed as input to the MMOX BoF and WG which is currently being chartered. It shows, that the main point of interoperability, namely the instantiation of avatars can be achieved with a very simple yet powerful protocol. It shall serve as a working showcase of light weight interoperation and as a reference point for

the discussion.

The protocol is Web/HTTP based, actually the transport is HTTP, the container format is XML and avatar data can use any MIME type. The architecture is inherently distributed by using basically XML to organize asset URLs.

The protocol has been used by Weblin for 2 years with 3 Mio. users and 25.000 concurrent clients. The primary purpose of the protocol is to enable "simulators" to instantiate ("rez") avatars of users they meet in a virtual world. The protocol has been designed for interoperability, specifically to be able to use avatars from closed virtual worlds on the Web. But it is equally suited to bridge from one world to another. Actually, the Web is regarded as just one virtual world with many regions, which are commonly called Web sites.

The protocol also offers a solution for the "foreign updates" problem, where the avatar changes in one world and the change should be displayed instantly in another world. Say your WoW avatar walks in SL and an effect times out. That should be visible in SL with minimum delay but also without maintaining long lists of subscriptions. The protocol has a lean way to communicate changes.

The protocol addresses the "the low hanging fruit" of simulator interoperability. But it is easily extensible, e.g. to virtual goods ("dragon heads"). It supports variants of avatar models if multiple formats are required. It provides trust using XML Signature (not specified here, but in use at Weblin). OAuth is the canonical choice for access authentication and selective disclosure of identity data.

It is assumed here, that the task of the MMOX WG will be restricted to interoperability between simulators and asset services, and that it will not include the communication between display and simulator. There are so many different models how worlds structure their communication between display and simulator. In Weblin the world simulation runs in the client. In many 3D worlds the simulator is regarded as a "server" which forwards scene changes to the display. Games like WoW do much more simulation in the client than Second Life. In the near future there will be simulation including rendering in the cloud with only thin displays. For this reason the MMOX WG should only work on simulator interoperability and keep the display-simulator communication out of the scope.

This document is identical to [VPTN3]. It has been reformatted and annotated for IETF submission. The document uses the term "virtual presence client" to refer to an entity, which interprets avatar data in order to instantiate avatars. This means any simulator, be it in a grid, in a cloud, or in a client.

Abstract

A virtual presence client needs information to display people who meet. It needs a name, an image, maybe an animated avatar, and more. This document describes the storage and exchange of public user identity data. The virtual presence identity data format is optimized for VP applications, where many people need the public data of their peers, some only once, some repeatedly, where changes happen frequently and must be propagated quickly with minimum bandwidth.

Table of Contents

1. Introduction

Any time users meet, they need at least a name to display their peer. Graphical client software needs more. It shows an image or an avatar. Clients also need other data for various purposes, e.g. availability

status, reputation, social status, a unique identifier, friends, inventory, communication addresses, and probably more data types in the future, than we can imagine now.

The data must be available to any peer. It must be available quickly. It should be cached to be re-used later, when people meet again. It must be updated quickly, if it changes, even if the change happens while there is no association between the clients. The data must always be up to date with a minimum use of bandwidth, because there are many users to meet and many changes to process.

Clients could subscribe for updates to be notified when user data changes. This would keep there local cache always up to date. Subscriptions are a perfect solution for instant messaging clients, where clients always show the latest information of peers on the roster (buddy list).

But in a VP application, client software needs the information only, if people meet. If user data changes, while users do not see each other, the change can go unnoticed. Actually it should not be propagated, because it will not be displayed anyway. The client software needs the latest information only if users meet. But then it the data must be available quickly with low overhead.

2. Concepts

This section describes the concepts of VP user data exchange. Basically, the VP user data exchange makes sure, that users see each other exactly like they want to be seen by their peers.

Users completely control their appearance. Users decide where they store their data. All the data, that makes up the appearance on other people's screens is summarized as the public "identity" of the user.

2.1 Identity Data

The identity usually has a nickname, an image, maybe a homepage URL, or communication addresses. It may contain or reference an electronic business card like a XMPP vCard. It may have an animated 3D model as avatar. It may even include a reference to a social network rating system.

It is up to the user to provide the data in the identity (identity provider). And it is up to the user who receives the identity data to display it (identity consumer). Clients are free to display the information they want to display.

Users can change their data any time. They can change the nickname, update the image, and any other item.

2.2 Identity Update

A simple way to communicate changes is a version number. The receiver stores the version number with the data. A new version number means, that the data changed. A unique digest of the data is very similar, but more general. The only requirement is, that the digest changes when the data changes. Actually a version number is a special kind of digest.

When users meet, then their clients exchange an identifier, which indicates the state of their user data. The identifier is a version number or a digest, or any other short text sequence, which identifies the state of the user data. If users change their data, e.g. the avatar image or a nickname, then they have to make sure, that their client communicates a different digest.

3. Specification

3.1 Identity Data

The identity is an XML document.

The top level identity-node contains multiple item-nodes.

Item-nodes either carry the item data as inner text or they reference external data.

In addition, item-nodes may have these attributes:

- id: a unique identifier of the item inside the identity
- contenttype: indicates how to use the item, e.g. "avatar"
- mimetype: data type, e.g. "image/gif"
- order: a number, which indicates the display preference, if there are multiple items of the same contenttype
- size: size of the item in bytes
- encoding: encoding of the node text, if any
- digest: a version identifier, which indicates the version of the item
- src: a URL, which points to the data.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<identity>
  <item id="avatar"
    contenttype="avatar"
    size="2579"
    order="1"
    src="http://avatar.foobar.com/astronaut.gif"
    digest="da03dc963a28b4dbccd28b3cad7a2177de41bad7"
```

```
      />
    <item id="properties"
      contenttype="properties"
      digest="cbe86cb69e979414d259e487ab500f138e42c6d7"
      mimetype="text/xml"
      >
      <![CDATA[<?xml version="1.0" encoding="UTF-8"?>
        <properties>
          <property name="Nickname" value="Tassadar"/>
          <property name="Gender" value="male"/>
        </properties>
      ]]>
  </identity>
```

3.1.1 External Data

External data is referenced by the src-attribute. The attribute value is a URL.

If the src-attribute is present, then the mimetype-attribute and the size-attribute may be used as hints, but the real values of data size and MIME-type are determined by the response when fetching the actual data.

3.1.2 Inline Data

If there is no src-attribute, then the item data is the inner text of the item-node.

The item data must be valid XML text. An optional encoding-attributes allows for base64-encoding of binary data.

3.1.3 Item Content Type

The contenttype-attribute indicates how the item is to be used. The client is free to ignore the attribute, but it helps to identify which item is to be shown as static image, which item contains the nickname, etc.

Possible values of the contenttype-attribute:

- "avatar": an static avatar image
- "properties": a list of key-value pairs
- ...

3.1.4 Item Order

An identity may contain multiple items of the same content type. There might be an "avatar"-item with mimetype="avatar/gif" and another one with mimetype="avatar/flash". Both need the appropriate

decoder software.

If a client has only one of the required avatar decoders, then it will usually select the item, that can be displayed rather than displaying none. But a client which has both decoders may use the order-attribute to determine which avatar is preferred by the user who provided the identity.

3.1.5 Item Encoding

Binary data must be encoded, if it is inline (inner text of the item-node).

Possible values of the encoding-attribute:

- "plain": not encoded (default)
- "base64": the data is base64 encoded. A decoder should dismiss embedded line breaks ("\r" and/or "\n"), tabs and white space.
- "URL": URL encoding with "&" and "=" separators, and %HH encoding of characters not allowed in HTTP-URLs.

3.1.6 Item Digest

Each item-node has a digest-attribute. The value of the digest-attribute must change, if the item data changes.

3.1.7 Properties

The "properties"-item contains short textual values. The item data is a list of key-value pairs.

The "properties"-item may have mimetype="text/xml".

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
  <property name="Nickname" value="Tassadar"/>
  <property name="Gender" value="male"/>
</properties>
```

Note: inline data must be inside a CDATA section.

Example:

```
<item id="properties"
  contenttype="properties"
  digest="cbe86cb69e979414d259e487ab500f138e42c6d7"
  mimetype="text/xml"
  >
```

```
<![CDATA[<?xml version="1.0" encoding="UTF-8"?>
```

```
        <properties>
          <property name="Nickname" value="Tassadar"/>
          <property name="Gender" value="male"/>
        </properties>
      ]]>
    </item>
```

Property keys currently used include:

- Nickname: a short label which may replace the nickname supplied by the chat protocol
- Gender: predefined values are "female", "male", "dontknow", "donttell", any other value allowed
- NicknameLink: a URL to be opened if people click the nickname
- Birthdate: free form date
- Profession: free form text
- Zodiaksign: predefined values are "Capricorn", "Aquarius", "Pisces", "Aries", "Taurus", "Gemini", "Cancer", "Leo", "Virgo", "Libra", "Scorpio", "Sagittarius", any other value allowed
- Eyecolor: free form text
- Country: ISO country code
- Languages: comma separated list of ISO country codes (e.g. "en") or language codes (e.g. "en_UK")
- Hobbies: free form text
- Interests: free form tags which describe private and professional interests
- Statement: short text message to the world
- Homepage: URL

All keys are optional.

Note: there are many free form text properties. They are meant for users, not for the machine. Standardized tags and/or a categorizations for hobbies, interests, profession, eye color, etc. may be defined separately using other keys or other identity content types.

Note: the "properties"-item may have a mimetype="text/plain". In this case the data is a line-feed separated list of key=value pairs. This variant is deprecated.

Example:

```
  Nickname=Tassadar
  Gender=male
```

3.1.8 Nickname

The nickname is very important, because clients will usually display

a nickname and an image. Since all chat protocols support the nickname natively, a nickname is always available.

The nickname of the identity may override the nickname supplied by the chat protocol.

The nickname is part of the item of contenttype="properties".

The nickname is not globally unique.

The nickname should not exceed 50 characters.

3.1.9 Avatar

An item-node of contenttype="avatar" contains the data to show an avatar image. The avatar may be of any type. There may be animated avatars

There may be multiple "avatar"-items.

At least one of the "avatar"-items should be an image. The "avatar"-image should be mimetype="image/gif", "image/jpeg", or "image/png". The dimensions should be should not exceed 100x100 pixel. The data size should not exceed 10 kB. All graphical clients should be able to display this "avatar"-image.

Note: clients may discover an alternate avatar-item of contenttype="avatar2". The "avatar2"-item has the same properties as the "avatar"-item.

3.1.10 Identity Digest

The identity digest is communicated to other clients.

The identity digest must change, if any of the items changes or if the list of items changes, e.g. if an item is added or removed.

The identity digest may be added to the identity-node as digest-attribute. This is not required by identity consumers. It makes the identity self-contained and simplifies identity-updates for identity providers.

Example:

```
<identity digest="888cf9a8f540a5d3ef240bbdc63db2fa12df0061">
  ...
</identity>
```

Note: the identity digest can be computed as a hash (message digest, e.g. MD5) of a concatenation of all item digests. 100 bytes should be

enough. It should not exceed 1 kB.

3.2 Identity Update

Clients communicate the identity of their users. They exchange the address of the identity file, the current identity digest, and a unique user ID. The details depend on the chat protocol.

Clients send and consume the identity triple:

- identity URL
- identity digest
- identity ID

A client, which receives such an identity-triple checks the identity digest of the user identified by the identity ID. If the digest is different from the one, that has been received earlier, then the client fetches the identity file using the identity URL.

After receiving the identity file, The client checks the item digest of all items for changes and fetches external item data, if required.

XMPP Example:

```
<presence ...(presence-attributes)>
  <x xmlns=" firebat:user:identity"
     id="http://foobar.myopenid.org"
     digest="888cf9a8f540a5d3ef240bbdc63db2fa12df0061"
     src="http://identity.virtual-presence.org/foobar"
  >
  ... (other child-nodes)
</presence>
```

In XMPP the identity is an extension node of the presence-node. The XML name space is "firebat:user:identity" ("Firebat" is the internal project name of a client). The name space will be changed in the future to follow XSF and IETF recommendations.

The example uses an OpenID as unique identity ID (id). Anything, that uniquely identifies a user will do.

The identity URL (src) points to the identity XML document.

3.2.1 Identity URL

The identity URL points to the identity document (3.1 "Identity Data").

The only URL scheme currently defined is "http".

The identity URL is mandatory.

3.2.2 Identity Digest

The identity digest is a short (compared to the identity data) textual identifier, which uniquely identifies a state of the identity data.

The identity digest might be a version number or a message digest of all item digests.

When users change their identity data, then they must take care, that the identity digest changes and that their client communicates the new identity digest.

The identity digest may be stored in the identity file as a digest-attribute of the identity-node.

The identity digest is optional, but clients should send digest and ID to enable caching of the identity data. Other clients may refuse to fetch identity data supplied without digest and ID.

3.2.3 Identity ID

The identity ID is the unique character string. The ID and the digest are used to cache the identity data.

The identity ID may be an email address, the hash of an email address, a URL, or any other globally unique character string.

Clients use the ID to associate an identity URL with a user, even if the identity URL changes. A user may change the identity URL (e.g. by changing the identity provider), but may still be recognized by other clients.

The identity digest is optional, but clients should send digest and ID to enable caching of the identity data. Other clients may refuse to fetch identity data supplied without digest and ID.

4. Requirements

This section lists the original requirements, which lead to the identity storage format described in this document.

4.1 Data Format

The format should be extensible to other data types.

The format should be extensible to new features.

The format should support multiple independent items.

The format should support hierarchical storage.

The format should be able to include items directly or by external reference.

The encoding should be simple so, that it can be written by users.

The encoding should be a very common data format.

The encoding should be able to embed other various types.

4.2 Caching and Updates

The format should enable item caching of individual items.

The format should enable cache updates for individual items.

Update notification should work with minimum overhead.

The data should be up to date very quickly if people meet.

4.3 Storage and Protocol

The storage should use a very common protocol, preferably HTTP, because HTTP is also the document and VPI protocol.

The storage should be distributed. There should not be a single storage server for user data.

The storage address should be a single address, e.g. a URL.

4.4 Ownership, Control, and Privacy

Users should completely control their appearance.

Users should be able to change the storage address quickly, if they move to another provider, without loosing their identity.

Users should be able to control their appearance.

Anonymous access of user data should be possible. In addition the can be personalized access to control the disclosure of information selectively.

5. IANA requests

This memo includes no request to IANA.

6. Security Considerations

6.1 Identity Data

The identity data is public. Anyone can access the data. There should

be a selective disclosure which differentiates between requesting

users.

6.2 Identity ID

While the identity ID is primarily used for caching, it can be used to identify users between visits. Clients may change identity ID, digest, and URL from time to time. But frequent changes render identity data caching useless. This would result in greatly increased traffic.

Rather, users might accept the fact, that their virtual presence actually makes them present to other users and that their peers may remember meeting them. The positive effects of meeting people and recognizing friends probably outweighs privacy implications, as they do in the real world.

7. References

7.1 Informative References

[VPTN3] http://www.virtual-presence.org/notes/VPTN-3.txt

Author's Address

Heiner Wolf
Waterloostr. 26
22769 Hamburg
Germany

Email: wolf.heiner@gmail.com