Internet Engineering Task Force
Internet Draft

IMPP WG
Jonathan Rosenberg
Dean Willis
Robert Sparks
Ben Campbell
dynamicsoft

Henning Schulzrinne
Jonathan Lennox
Columbia U.

Bernard Aboba
Christian Huitema
David Gurle
Microsoft

draft-rosenberg-impp-pidf-00.txt
June 15, 2000
Expires: December, 2000

A Data Format for Presence Using XML

STATUS OF THIS MEMO

Abstract

This document describes an XML based data format for conveying presence information. The format is one instantiation of an abstract presence data model also described here.

1 Introduction

Presence is (indirectly) defined in RFC2778 [1] as subscription to and notification of changes in the communications state of a user. This communications state consists of the set of communications means, communications address, and status of that user. A presence document is a computer readable piece of data that contains this presence information. Presence documents are conveyed by a presence protocol [2], but may also be conveyed out of bands. The presence documents are useful and complete in and of themselves, and do not

rely on information conveyed by their transfer means in order to be useful.

It is fundamental to our notion of presence that a single presentity is represented by a multitude of presence user agents (PUAs), each of which generates presence information for a particular subset of the overall presence state of a presentity. This results in the requirement of a definition of an abstract presence data model, which defines how these presence components are combined to yield a complete presence document. This data model is, in fact, independent of the presence data format itself. However, we describe it here so that the document is complete.

This presence data format is based on XML. XML has its strengths and weaknesses, and for this reason, we actually define an additional presence data format which is more suited to smaller footprint endpoints [3].

2 Presence Data Model

We define presence as a set of atoms. Each atom defines a component of the presence state of a presentity. The presence state of the presentity is defined by taking the union of the current set of atoms. The process of combining presence information from different sources operates on the granularity of atoms; the content of atoms is not examined when performing such merging. Any presence data format MUST define a means for unioning which can be accomplished through syntactic understanding alone of the presence document. A presence data format MAY define a means for unioning based on semantic understanding.

Each atom MUST contain a field which defines the presentity for whom the atom represents. This is in support of Requirement 3.1.2 of RFC2779 [4], which defines requirements for presence and instant messaging.

Each atom has an opaque identifier, uniquely identifying it. There is no meaning associated with these identifiers. Two atoms are for the same piece of presence state when they identify the same presentity, and when their atoms match through a byte-wise equality check of their identifiers.

Each atom also has an expiration time, indicating the lifetime of the data contained within the atom.

A subscriber will receive a set of atoms representing the presence for a user. Computation of the complete presence state of that presentity proceeds as follows:

- The subscriber records, for each atom, the most recent instance of that atom. Most recent is *not* defined by expiration time, but rather through sequencing provided by the transfer protocol, or temporal ordering of receipt or fetch, if no such sequencing is provided.

- If the most recent atom has an expiration time earlier than the current time, that atom is removed completely from the list of atoms contributing to presence.

- The subscriber generates the final piece of presence by taking the set of unexpired atoms for the same presentity, and unioning them as defined by the presence format.

This fairly simple model of presence data is very flexible. It supports several different usage scenarios:

- Presence data can be generated by a single entity, but be transferred only as updates. Each piece of the presence being updated is a single atom. The presentity need only send an atom when it changes, or send an update before it expires.

- Presence data can be generated by a single entity, and the entire presence state is transferred on each notification.

- Presence data can be generated by several entities, each updating a set of atoms for the presentity.

3 Applying the Model to the Presence Protocol

When presence documents corresponding to this model are transferred by the presence protocol [2], two issues need to be considered.

First, the presence protocol [2] provides sequencing of notifications, through the CSeq header. When two notifications come from the same source (the To, From, and Call-ID are the same), the presence data MAY be ordered based on the CSeq ordering. If the notifications come from different sources, the temporal order of delivery is used to determine ordering.

Secondly, it is RECOMMENDED that each atom correspond to a single Contact address in the registrations. Furthermore, the atom identifier SHOULD be computed as the MD5 hash of the URI of that Contact. This allows both server and PUA to know the identifier, so that they can alternately take over generation of updates for the same atom.

4 Components of the Document

The presence document format defined here is based on XML, and is similar in concept to the proposed PIDF format by Vermeulen [5]. Usage of XML brings several benefits; as presence data is likely to be rendered, the ability to define XSL and CSS documents for displaying of presence is useful. The ability to digitally sign subcomponents of an XML document is also useful.

The presence document always contains the top level element "presence," which indicates that the remainder of the document contains presence information.

```
<!ELEMENT presence (presentity, atom*)>
```

The first sub-element of the presence element is the "presentity" element, which identifies the presentity for whom the presence data is being reported.

```
<!ELEMENT presentity (#PCDATA)>
<!ATTLIST presentity uri CDATA #REQUIRED>
```

The presentity tag has a single mandatory attribute, uri, which gives the address of the presentity. The content of the presentity tag is parsed character data giving a human-readable name.

This parsed character data may consist of plain text. It may also contain XML mark-up from an XML document type designed to present human-readable content, such as XHTML [6], MathML [7], SVG [8], VoiceXML [9], SMIL [10], etc., when properly marked with an XML namespace identifier. A program interpreting a buddy list SHOULD interpret and display XML markup from an unknown or unsupported XML namespace as it would a document of type "text/xml" with an unknown or unsupported document type declaration.

> Note that some of the items in this list of document formats are deliberately rather over-the-top; it seems improbable that having a SMIL presentation to describe a presentity would actually be useful. XHTML, however, will likely be common.

Following the presentity tag within the presence tag is a list of atoms.

4.1 Atoms

Atoms are structured as a collection of addresses. These can either be communications addresses, represented by URLs, or a postal address.

```
<!ELEMENT atom (postal?, address*)>
<!ATTLIST atom atomid  CDATA #REQUIRED
               expires CDATA #IMPLIED>
```

The atom element has the mandatory attribute "id", the unique identifier for the group, and the optional attribute "expires" which indicates the time after which the presence data should be considered invalid. The expiration time is expressed as an integral number of seconds since January 1, 1970, 00:00 UTC.

A postal address is indicated by the "postal" element, and consists of freeform text:

```
<!ELEMENT postal (#PCDATA)>
```

It may contain XML markup from some external namespace, as described previously.

> It would be nice to require an XML format for postal addresses. Does any exist? vCard XML profile or something?

Communications addresses are described by the "address" element.

```
<!ELEMENT address (status | class | duplex | feature | note)*>
<!ATTLIST address uri      CDATA  #REQUIRED
                  priority CDATA  #IMPLIED>
```

The "address" element has a single mandatory attribute, uri, which gives the URI of the communications address being described. It also

has an optional attribute "priority". The priority tag contains an integer that indicates the relative preference of this address over other addresses. It is a floating-point value between 0 and 1, with 1 being the highest preference.

Within the address tag, several subtags are defined to specify characteristics of the communications address. These tags have the following meanings:

status: Status is an indicator, meant for machine consumption, which indicates the status of this communications address. Valid values are "open", which means communications can be attempted to this address, "closed", which means communications cannot be attempted, and "inuse", which means that communications means is currently being actively used with the entity receiving the presence document. For example, if an instant messaging URL is placed in the uri attribute of the address, and the status is "inuse", this means that the user sending the updated presence document is currently typing an instant message to the recipient of the presence document.

This enables a recent feature on MSN, which allows you to see when the person you are IMing is currently typing a reply to your IM.

```
<!ELEMENT status EMPTY>
<!ATTLIST status status (open|closed|inuse) #REQUIRED>
```

class: The class tag contains either the value "business" or "personal", indicating whether the address is for business or non-business use. There can only be one class tag per address.

```
<!ELEMENT class EMPTY>
<!ATTLIST class class   (business|personal) #REQUIRED>
```

duplex: The duplex tag contains one of the values "full", "half", "send-only" and "receive-only". It indicates

whether the address can be used for communications in one direction, the other direction, or both. For example, a page would be considered receive-only. There can only be one duplex tag per address.

```
<!ELEMENT duplex EMPTY>
<!ATTLIST duplex duplex (full|half|send-only|receive-only)
#REQUIRED>
```

feature: The feature tag lists features specific to that communications means. For voice addresses, defined values include "voice-mail" and "attendant". There can be more than one feature tag per address.

```
<!ELEMENT feature EMPTY>
<!ATTLIST feature feature (voicemail|attendant) #REQUIRED>
```

mobility: The mobility tag indicates whether the terminal with the given communications address is moving around ("mobile") or fixed ("fixed"). There can only be a single mobility tag per address.

```
<!ELEMENT mobility EMPTY>
<!ATTLIST mobility mobility (fixed|mobile) #REQUIRED>
```

note: Note contains freeform text meant for display to the user, indicating some kind of information about the communications address. There can only be one note tag per address. The note tag may contain XML data from a properly-qualified external XML namespace.

```
<!ELEMENT note (#PCDATA)>
```

A PIDF document which appears as a top-level XML document is

identified with the formal public identifier "-//IETF//DTD RFCxxxx XPIDF 1.0//EN". If this document is published as an RFC, "xxxx" will be replaced by the RFC number. PIDF documents have the MIME type "application/xpidf+xml".

> The "+xml" component of the type name conforms with the format of XML MIME types introduced by Murata et al [11]; this allows XML-aware but PIDF-unaware rendering tools to display PIDF usefully.

A PIDF document embedded as a fragment within another XML document is identified with the XML namespace identifier "http://www.ietf.org/internet-drafts/draft-rosenberg-impp-pidf-00.txt". If this document is published as an RFC, the namespace identifier will be "http://www.rfc-editor.org/rfc/rfcxxxx.txt", where xxxx is the RFC number.

> Note that the URIs specifying XML namespaces are only globally unique names; they do not have to reference any particular actual object. The URI of a canonical source of this specification meets the requirement of being globally unique, and is also useful to document the format.

5 Constructing the union of two presence documents

Construction of the union of two presence documents is straightforward. Two presence documents can be unioned only if they both refer to the same presentity. The atoms from each presence document is extracted. If some number of atoms have the same id, the most recent atom is selected. Most recent is defined either by the transfer protocol, or through temporal ordering. A new presence document is then constructed, using the set of atoms obtained from the presence documents. For example, consider presence document A:

```
<?xml version="1.0"?>
<!DOCTYPE presence
   PUBLIC "-//IETF//DTD RFCxxxx XPIDF 1.0//EN" "xpidf.dtd">
<presence>
  <presentity uri="sip:user@example.com;method=SUBSCRIBE" />
  <atom atomid="779js0a98">
    <address uri="sip:user@example.com">
      <status status="open" />
    </address>
  </atom>
</presence>
```

and presence document B:

```
<?xml version="1.0"?>
<!DOCTYPE presence
   PUBLIC "-//IETF//DTD RFCxxxx XPIDF 1.0//EN" "xpidf.dtd">
<presence>
  <presentity uri="sip:user@example.com;method=SUBSCRIBE" />
  <atom atomid="22">
    <address uri="mailto:user@example.com">
      <status status="open" />
    </address>
  </atom>
</presence>
```

The combined document looks like:

```
<?xml version="1.0"?>
<!DOCTYPE presence
   PUBLIC "-//IETF//DTD RFCxxxx XPIDF 1.0//EN" "xpidf.dtd">
<presence>
  <presentity uri="sip:user@example.com;method=SUBSCRIBE" />
  <atom atomid="779js0a98">
    <address uri="sip:user@example.com">
      <status status="open" />
    </address>
  </atom>
  <atom atomid="22">
    <address uri="mailto:user@example.com">
      <status status="open" />
    </address>
  </atom>
</presence>
```

6 Example

The following is an example presence document:

```
<?xml version="1.0"?>
<!DOCTYPE presence
   PUBLIC "-//IETF//DTD RFCxxxx XPIDF 1.0//EN" "xpidf.dtd">
<presence>
```

```
    <presentity uri="sip:user@example.com;method=SUBSCRIBE" />
    <atom atomid="779js0a98">
      <address uri="sip:user@example.com" priority="0.8">
        <status status="open" />
        <duplex duplex="full" />
        <feature feature="voicemail" />
        <feature feature="attendant" />
      </address>
      <address uri="mailto:user@example.com">
        <status status="open" />
        <note>Send email if I'm not around</note>
      </address>
    </atom>
  </presence>
```

7 Mapping from SIP Registrations

This section discusses how one would map the contents of a SIP REGISTER message into a presence document.

Typically, each Contact header in the registration will be a separate atom. That atom will have a single URL, which is the URL from the Contact header. If the registration is not expired, the status is set to open. Once the registration expires, the address is set to closed, and can be removed completely from the presence document at some point in the future.

If the UA sending the registration supports caller preferences [12], and has included them in the contact header, they are mapped one to one into their equivalent XML tags described here.

8 DTD

The following is the DTD for the presence document.

```
<?xml version="1.0" encoding="US-ASCII" ?>

<!ELEMENT presence (presentity, atom*)>

<!ELEMENT presentity (#PCDATA)>
<!ATTLIST presentity uri CDATA #REQUIRED>

<!ELEMENT atom (postal?, address*)>
<!ATTLIST atom atomid  CDATA #REQUIRED
               expires CDATA #IMPLIED>
```

```
<!ELEMENT postal (#PCDATA)>

<!ELEMENT address (status | class | duplex | feature | note)*>
<!ATTLIST address uri      CDATA  #REQUIRED
                  priority CDATA  #IMPLIED>

<!ELEMENT status EMPTY>
<!ATTLIST status status (open|closed|inuse) #REQUIRED>

<!ELEMENT class EMPTY>
<!ATTLIST class class   (business|personal) #REQUIRED>

<!ELEMENT duplex EMPTY>
<!ATTLIST duplex duplex (full|half|send-only|receive-only) #REQUIRED>

<!ELEMENT feature EMPTY>
<!ATTLIST feature feature (voicemail|attendant) #REQUIRED>

<!ELEMENT mobility EMPTY>
<!ATTLIST mobility mobility (fixed|mobile) #REQUIRED>

<!ELEMENT note (#PCDATA)>
```

9 Evaluation against Requirements

The following section indicates how this proposal meets the requirements outlined in RFC2779 [4].

Requirement 3.1.2: The common presence format MUST include a means to uniquely identify the PRESENTITY whose PRESENCE INFORMATION is reported. This is supported through the presentity tag.

Requirement 3.1.3: The common presence format MUST include a means to encapsulate contact information for the PRESENTITY's PRINCIPAL (if applicable), such as email address, telephone number, postal address, or the like. This is supported through the postal and address tags.

Requirement 3.1.4: There MUST be a means of extending the common presence format to represent additional information not included in the common format, without undermining or rendering invalid the fields of the common format This is supported through the definitions of new tags for the XML

presence document.

Requirement 3.1.5: The working group must define the extension and registration mechanisms for presence information schema, including new STATUS conditions and new forms for OTHER PRESENCE MARKUP. This is readily accomplished, although not specified in this version of the document.

Requirement 3.1.6: The presence format SHOULD be based on IETF standards such as vCard [RFC 2426] if possible. The format is based on XML, a standard structured data representation used in many IETF standards, and on URLs, also defined by IETF.

Requirement 4.4.1: The common presence format MUST define a minimum standard presence schema suitable for INSTANT MESSAGE SERVICES. This is supported through a URL corresponding to an instant message. This URL, as proposed in [13] is of the form sip:user@host;method=SUBSCRIBE. This URL is then included in an address tag.

Requirement 4.4.2: When used in a system supporting INSTANT MESSAGES, the common presence format MUST include a means to represent the STATUS conditions OPEN and CLOSED. These statuses are defined here.

Requirement 4.4.3: The STATUS conditions OPEN and CLOSED may also be applied to messaging or communication modes other than INSTANT MESSAGE SERVICES. The presence document here applies statuses to any communications means. IM is not treated differently than any other.

10 Authors Addresses

Jonathan Rosenberg
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: jdrosen@dynamicsoft.com

Dean Willis

dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: dwillis@dynamicsoft.com

Robert Sparks
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: rsparks@dynamicsoft.com

Ben Campbell
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: bcampbell@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

Jonathan Lennox
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: lennox@cs.columbia.edu

Christian Huitema
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
email: huitema@microsoft.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
email: bernarda@microsoft.com

David Gurle
Microsoft Corporation

One Microsoft Way
Redmond, WA 98052-6399
email: dgurle@microsoft.com

11 Bibliography

[1] M. Day, J. Rosenberg, and H. Sugano, "A model for presence and instant messaging," Request for Comments 2778, Internet Engineering Task Force, Feb. 2000.

[2] J. Rosenberg, R. Sparks, D. Willis, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, and D. Gurle, "SIP extensions for presence," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

[3] J. Rosenberg, R. Sparks, D. Willis, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, and D. Gurle, "A lightweight presence information data format (LPIDF)," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

[4] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, "Instant messaging / presence protocol requirements," Request for Comments 2779, Internet Engineering Task Force, Feb. 2000.

[5] C. Vermeulen, "Presence info data format (PIDF)," Internet Draft, Internet Engineering Task Force, Dec. 1999. Work in progress.

[6] W. H. W. Group, "XHTML 1.0: The extensible hypertext markup language," W3C Recommendation REC-xhtml1-20000126, World Wide Web Consortium (W3C), Jan. 2000. Available at http://www.w3.org/TR/xhtml1/.

[7] P. Ion and R. Miner, "Mathematical markup language (MathML) 1.01 specification," W3C Recommendation REC-MathML-19990707, World Wide Web Consortium (W3C), July 1999. Available at http://www.w3.org/TR/REC-MathML/.

[8] J. Ferraiolo, "Scalable vector graphics (SVG) 1.0 specification," W3C Working Draft WD-SVG-20000303, World Wide Web Consortium (W3C), Mar. 2000. Available at http://www.w3.org/TR/SVG/.

[9] VoiceXML Forum, "Voice extensible markup language (VoiceXML) version 1.0," W3C Note NOTE-voicexml-20000505, World Wide Web Consortium (W3C), May 2000. Available at http://www.w3.org/TR/voicexml/.

[10] P. Hoschka, "Synchronized multimedia integration language (SMIL) 1.0 specification," W3C Recommendation REC-smil-19980615, World Wide Web Consortium (W3C), June 1998. Available at http://www.w3.org/TR/REC-smil/.

[11] M. Murata and S. S. Laurent, "XML media types," Internet Draft, Internet Engineering Task Force, Jan. 2000. Work in progress.

[12] H. Schulzrinne and J. Rosenberg, "SIP caller preferences and callee capabilities," Internet Draft, Internet Engineering Task Force, Mar. 2000. Work in progress.

[13] J. Rosenberg, R. Sparks, D. Willis, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, and D. Gurle, "SIP extensions for instant messaging," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.