```
MESSAGE sip:user1@user1pc.domain.com SIP/2.0
Via: SIP/2.0/UDP user2pc.domain.com
To: im:user1@domain.com
From: im:user2@domain.com;tag=ab8asdasd9
Contact: sip:user2@user2pc.domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: multipart/signed; boundary=next;
              MDALG=SHA-1; type=application/pkcs7
Content-Length: <however many bytes that is below>

--next
Content-Type: message/cpim

From: <im:user2@domain.com>
To: <im:user1@domain.com>
Date: 2001-02-28T01:20:00-06:00

Content-Type: text/plain

My name is User2, not Watson.

--next
Content-Type: application/pkcs7

(signature stuff)
 :
--next--
```

This is sent directly to user1, who responds with a 200 OK in message F6:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP user2pc.domain.com
To: im:user1@domain.com;tag=2c09sj3sd9
From: im:user2@domain.com;tag=ab8asdasd9
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Length: 0
```

9. Open Issues

9.1 Must a MESSAGE actually include a message?

Section 6 specifies that a MESSAGE MAY contain a MIME body. Should this be MUST? Does it make sense to have a MESSAGE with no body?

9.2 Should support for message/cpim be mandatory in all UAs?

Section 6 requires that UAs implementing MESSAGE support text/plain

bodies as the lowest common denominator. Should this be message/cpim instead? Any UA wishing to support end-end signing or encryption of messages passing across simple/apex/prim boundaries MUST support message/cpim. If, however, end-end security is not desired, clients and messaging can be made a little lighter by not including the message/cpim wrapper. An unsigned message/cpim body can be created from messages from those clients when crossing a boundary that requires one.

9.3 message/cpim and the Accept header

Do we need text to make it clear that a UA should indicate the mime types it supports _inside_ a message/cpim body as well as supporting message/cpim?

9.4 Message Sessions

Several implementations of the -00 version of this draft grouped messages in a common thread by placing them in a "call-leg" (common To, From, and Call-ID). The first message sent or received in a thread established the leg. This has provided enough information to allow user interfaces to present separate threads in separate dialogs. There is some concern that there is no way to formally terminate this "call-leg".

The -00 version noded that there is state at the UA associated with this notion of session, encapsulated in the Call-ID, Route headers, and CSeq numbers. A UA MAY terminate this session at any time, including after each MESSAGE. No messaging is required to terminate it. Any associated state with the session is simply discarded. The idempotency of SIP requests will ensure that if one side (side A) discards session state, and the other (side B) does not, a message from side B will appear as a new IM, and standard processing will reconstitute the session on side A.

- Should we define a way to use INVITE/BYE to surround a group of MESSAGE requests that are part of a logical session?

9.5 What would a body in a 200 OK to a MESSAGE mean?

Section 6.5 states "A 200 class response to a MESSAGE request MAY contain a body, but this will often not be the case, since these responses are generated automatically." If one were to appear, what would it mean?

9.6 The im: URL and RFC2543 proxies and registrars

What are the implications of an im: URL showing up in the request URI in a MESSAGE request received by an RFC2543 proxy, or the To: header of a REGISTER request received by an RFC2543 registrar?

9.7 Providing im: URL in Contact headers

What are the ramifications of a UA providing an im: URL in a Contact: header for a REGISTER method, or a MESSAGE method? For the forseeable future, most SIP endpoints aren't going to have SRV records of the form _im._sip.host or even _sip.host pointing to them. Falling back to A records in that case seems to preclude the use of non-UDP transports.

9.8 Congestion control

Per the amendments made to the SIMPLE charter by the IESG prior to approval, congestion control needs attention. In particular the requirements of BCP 41 must be met by this extension. Specifying the use of transport protocols with congestion control built in, particularly with the recommendation of reuse of connections, is an option. The question is when can we use those that don't (UDP) and what needs to be done in addition to what SIP already does in that case. Among other things, this interacts with Section 9.7

9.9 Mapping to CPIM

This document needs to detail the mapping of this extension onto CPIM.

10. Acknowledgements

The authors would like to thank the following people for their support of the concept of SIP for IM, support for this work, and for their useful comments and insights:

Jon Peterson	Level(3) Communications
Sean Olson	Ericsson
Adam Roach	Ericsson
Billy Biggs	University of Waterloo
Stuart Barkley	UUNet
Mauricio Arango	SUN
Richard Shockey	Shockey Consulting LLC
Jorgen Bjorker	Hotsip
Henry Sinnreich	MCI Worldcom
Ronald Akers	Motorola

References

[1] DellaFera, C. A., Eichin, M. W., French, R. S., Jedlinski, D. C., Kohl, J. T. and W. E. Sommerfeld, "The Zephyr notification service", in USENIX Winter Conference (Dallas, Texas), Feb. 1988.

[2] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[3] Day, M., Aggarwal, S. and J. Vincent, "Instant Messaging /

Presence Protocol Requirements", RFC 2779, February 2000.

[4] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[5] Rosenberg, J. and H. Schulzrinne, "SCTP as a transport for SIP", draft-rosenberg-sip-sctp-00 (work in progress), June 2000.

[6] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[7] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[8] Rosenberg, J. and H. Schulzrinne, "SIP caller preferences and callee capabilities", draft-ietf-sip-callerprefs-03 (work in progress), November 2000.

[9] Donovan, S., "The SIP INFO Method", RFC 2976, October 2000.

[10] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[11] Dierks, T., Allen, C., Treese, W., Karlton, P. L., Freier, A. O. and P. C. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[12] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[13] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[14] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[15] Crocker, D., Diacakis, A., Mazzoldi, F., Huitema, C., Klyne, G., Rose, M., Rosenberg, J., Sparks, R. and H. Sugano, "A Common Profile for Instant Messaging (CPIM)", draft-ietf-impp-cpim-01 (work in progress), February 2001.

[16] Atkins, D. and G. Klyne, "Common Presence and Instant Messaging Message Format", draft-ietf-impp-cpim-msgfmt-00 (work in progress), February 2001.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
200 Executive Drive

Suite 120
West Orange, NJ 07052

email: jdrosen@dynamicsoft.com

Dean Willis
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

email: dwillis@dynamicsoft.com

Robert J. Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

email: rsparks@dynamicsoft.com

Ben Cambpell
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

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

David Oran
Cisco Systems
170 West Tasman Dr.
San Jose, CA 95134

email: oran@cisco.com

Appendix A. Requirements Evaluation

This section was moved forward verbatim from -00.

RFC 2779 [3] outlines requirements for IM and presence protocols. The document describes both shared requirements and IM and presence specific requirements. Examining each of the IM requirements in turn, we also observe that they are met by this proposal:

"Requirement 2.1.1: The protocols MUST allow a PRESENCE SERVICE to be available independent of whether an INSTANT MESSAGE SERVICE is available, and vice-versa." This requirement is met by the separation of presence and IM which we propose here.

"Requirement 2.1.2. The protocols must not assume that an INSTANT INBOX is necessarily reached by the same IDENTIFIER as that of a PRESENTITY. Specifically, the protocols must assume that some INSTANT INBOXes may have no associated PRESENTITIES, and vice versa." This requirement is also easily met by any architecture which completely separates IM and presence as we propose.

"Requirement 2.1.3. The protocols MUST also allow an INSTANT INBOX to be reached via the same IDENTIFIER as the IDENTIFIER of some PRESENTITY." Same as above.

"Requirement 2.1.4. The administration and naming of ENTITIES within a given DOMAIN MUST be able to operate independently of actions in any other DOMAIN." This requirement is met by SIP. SIP uses email-like identifiers which consist of a user name at a domain. Administration of user names is done completely within the domain, and these user names have no defined rules or organization that needs to be known outside of the domain in order for SIP to operate.

"Requirement 2.1.5. The protocol MUST allow for an arbitrary number of DOMAINS within the NAMESPACE." This requirement is met by SIP. SIP uses standard DNS domains, which are not restricted in number.

"Requirement 2.2.1. It MUST be possible for ENTITIES in one DOMAIN to interoperate with ENTITIES in another DOMAIN, without the DOMAINS having previously been aware of each other." This requirement is met by SIP, as it is essential for establishing sessions as well. DNS SRV records are used to discover servers for a particular service within a domain. They are a generalization of MX records, used for email routing. SIP defines procedures for usage of DNS records to find servers in another domains, which include SRV lookups. This allows domains to communicate without prior setup.

"Requirement 2.2.2: The protocol MUST be capable of meeting its other functional and performance requirements even when there are millions of ENTITIES within a single DOMAIN." Whilst it is hard to judge whether this can be met by examining the architecture of a protocol, SIP has numerous mechanisms for achieving large scales of users within a domain. It allows hierarchies of servers, whereby the namespace can be partitioned among servers. Servers near the top of the hierarchy, used solely for routing, can be stateless, providing excellent scale.

"Requirement 2.2.3: The protocol MUST be capable of meeting its other functional and performance requirements when there are millions of DOMAINS within the single NAMESPACE." The usage of DNS for dividing the namespace into domains provides the same scale as todays email systems, which support millions of DOMAINS.

"Requirement 2.3.5: The PRINCIPAL controlling an INSTANT INBOX MUST be able to control which other PRINCIPALS, if any, can send INSTANT MESSAGES to that INSTANT INBOX." This is provided by access control mechanisms, outside the scope of this extension.

"Requirement 2.3.6: The PRINCIPAL controlling an INSTANT INBOX MUST be able to control which other PRINCIPALS, if any, can read INSTANT MESSAGES from that INSTANT INBOX." This is accomplished through authenticated registration requests. Registrations are used to determine which user gets delivered an instant message. Policy in proxies can allow only certain users to register

contact address for a particular inbox (an inbox is defined by the address-of- record in the To field in the registration).

"Requirement 2.4.3: The protocol MUST allow the sending of an INSTANT MESSAGE both directly and via intermediaries, such as PROXIES." This is fundamental to the operation of SIP.

"Requirement 2.4.4: The protocol proxying facilities and transport practices MUST allow ADMINISTRATORS ways to enable and disable protocol activity through existing and commonly-deployed FIREWALLS. The protocol MUST specify how it can be effectively filtered by such FIREWALLS." Although SIP itself runs on port 5060 by default, any other port can be used. It is simple to specify that IM should run on a different port, if so desired.

"Requirement 2.5.1. The protocol MUST provide means to ensure confidence that a received message (NOTIFICATION or INSTANT MESSAGE) has not been corrupted or tampered with." This is supported by SIPs PGP and S/MIME authentication mechanism.

"Requirement 2.5.2. The protocol MUST provide means to ensure confidence that a received message (NOTIFICATION or INSTANT

MESSAGE) has not been recorded and played back by an adversary." This is provided by SIP's challenge response authentication mechanisms, through timestamp-based replay prevention, or through stateful storage of previous transaction identifiers (the combination of To, From, Call-ID, CSeq).

"Requirement 2.5.3. The protocol MUST provide means to ensure that a sent message (NOTIFICATION or INSTANT MESSAGE) is only readable by ENTITIES that the sender allows." This is supported through SIPs end to end and hop by hop encryption mechanisms.

"Requirement 2.5.4. The protocol MUST allow any client to use the means to ensure non-corruption, non-playback, and privacy, but the protocol MUST NOT require that all clients use these means at all times." All algorithms for security in SIP are optional.

"Requirement 4.1.1. All ENTITIES sending and receiving INSTANT MESSAGES MUST implement at least a common base format for INSTANT MESSAGES." We specify text/plain here.

"Requirement 4.1.2. The common base format for an INSTANT MESSAGE MUST identify the sender and intended recipient." This is accomplished with the To and From fields in SIP.

"Requirement 4.1.3. The common message format MUST include a return address for the receiver to reply to the sender with another INSTANT MESSAGE." This is done through the Contact headers defined in SIP.

"Requirement 4.1.4. The common message format SHOULD include standard forms of addresses or contact means for media other than

INSTANT MESSAGES, such as telephone numbers or email addresses." SIP supports any URL format in the Contact headers. Furthermore, the body of a MESSAGE request can be multipart, and contain things like vCards.

"Requirement 4.1.5. The common message format MUST permit the encoding and identification of the message payload to allow for non-ASCII or encrypted content." MIME content labeling is used in SIP.

"Requirement 4.1.6. The protocol must reflect best current practices related to internationalization." SIP uses UTF-8 and is completely internationalized.

"Requirement 4.1.7. The protocol must reflect best current practices related to accessibility." Additional requirements are needed on what is required for accessibility.

"Requirement 4.1.9. The working group MUST determine whether the common message format includes fields for numbering or identifying messages. If there are such fields, the working group MUST define the scope within which such identifiers are unique and the acceptable means of generating such identifiers." This is done with the combination of Call-ID and CSeq. The mechanisms for guaranteeing uniqueness are specified in SIP.

"Requirement 4.1.10. The common message format SHOULD be based on IETF-standard MIME (RFC 2045)[14]." SIP uses MIME.

"Requirement 4.2.1. The protocol MUST include mechanisms so that a sender can be informed of the SUCCESSFUL DELIVERY of an INSTANT MESSAGE or reasons for failure. The working group must determine what mechanisms apply when final delivery status is unknown, such as when a message is relayed to non-IMPP systems." SIP specifies notification of successful delivery through 200 OK. When delivery of requests through gateways, success can be indicated only through the SIP component (if the gateway acts as a UAS/UAC) or through the entire system (if it acts like a proxy).

"Requirement 4.3.1. The transport of INSTANT MESSAGES MUST be sufficiently rapid to allow for comfortable conversational exchanges of short messages." The support for end to end messaging (i.e., without intervening proxies) allows IMs to be delivered as rapidly as possible. The UDP reliability mechanisms also support fast recovery from loss.

Full Copyright Statement

Acknowledgement

Funding for the RFC editor function is currently provided by the Internet Society.

Internet Engineering Task Force
Internet Draft
draft-ietf-simple-presence-00.txt
March 30, 2001
Expires: September 2001

SIMPLE WG
Rosenberg et al.
Various Places

SIP Extensions for Presence

STATUS OF THIS MEMO

Abstract

This document proposes an extension to SIP for subscriptions and notifications of user presence. User presence is defined as the willingness and ability of a user to communicate with other users on the network. Historically, presence has been limited to "on-line" and "off-line" indicators; the notion of presence here is broader. Subscriptions and notifications of user presence are supported by defining an event package within the general SIP event notification framework. This protocol is also compliant with the Common Presence and Instant Messaging (CPIM) framework.

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 protocol is a protocol for providing such a service over the Internet or any IP network.

This document proposes an extension to the Session Initiation Protocol (SIP) [2] for presence. This extension is a concrete instantiation of the general event notification framework defined for SIP [3], and as such, makes use of the SUBSCRIBE and NOTIFY methods defined there. User presence is particularly well suited for SIP. SIP registrars and location services already hold user presence information; it is uploaded to these devices through REGISTER messages, and used to route calls to those users. Furthermore, SIP networks already route INVITE messages from any user on the network to the proxy that holds the registration state for a user. As this state is user presence, those SIP networks can also allow SUBSCRIBE requests to be routed to the same proxy. This means that SIP networks can be reused to establish global connectivity for presence subscriptions and notifications.

This extension is based on the concept of a presence agent, which is a new logical entity that is capable of accepting subscriptions, storing subscription state, and generating notifications when there are changes in user presence. The entity is defined as a logical one, since it is generally co-resident with another entity, and can even move around during the lifetime of a subscription.

This extension is also compliant with the Common Presence and Instant Messaging (CPIM) framework that has been defined in [4]. This allows SIP for presence to easily interwork with other presence systems compliant to CPIM.

2 Definitions

This document uses the terms as defined in [1]. Additionally, the following terms are defined and/or additionally clarified:

Presence User Agent (PUA): A Presence User Agent manipulates presence information for a presentity. In SIP terms, this means that a PUA generates REGISTER requests, conveying some kind of information about the presentity. We explicitly allow multiple PUAs per presentity. This means that a user can have many devices (such as a cell phone and PDA), each of which is independently generating a component of the overall presence information for a presentity. PUAs push data into the presence system, but are outside of it, in that they do not receive SUBSCRIBE messages, or send NOTIFY.

Presence Agent (PA): A presence agent is a SIP user agent which is capable of receiving SUBSCRIBE requests, responding to them, and generating notifications of changes in presence state. A presence agent must have complete knowledge of the

presence state of a presentity. Typically, this is accomplished by co-locating the PA with the proxy/registrar, or the presence user agent of the presentity. A PA is always addressable with a SIP URL.

Presence Server: A presence server is a logical entity that can act as either a presence agent or as a proxy server for SUBSCRIBE requests. When acting as a PA, it is aware of the presence information of the presentity through some protocol means. This protocol means can be SIP REGISTER requests, but other mechanisms are allowed. When acting as a proxy, the SUBSCRIBE requests are proxied to another entity that may act as a PA.

Presence Client: A presence client is a presence agent that is colocated with a PUA. It is aware of the presence information of the presentity because it is co-located with the entity that manipulates this presence information.

3 Overview of Operation

In this section, we present an overview of the operation of this extension.

When an entity, the subscriber, wishes to learn about presence information from some user, it creates a SUBSCRIBE request. This request identifies the desired presentity in the request URI, using either a presence URL or a SIP URL. The subscription is carried along SIP proxies as any other INVITE would be. It eventually arrives at a presence server, which can either terminate the subscription (in which case it acts as the presence agent for the presentity), or proxy it on to a presence client. If the presence client handles the subscription, it is effectively acting as the presence agent for the presentity. The decision about whether to proxy or terminate the SUBSCRIBE is a local matter; however, we describe one way to effect such a configuration, using REGISTER.

The presence agent (whether in the presence server or presence client) first authenticates the subscription, then authorizes it. The means for authorization are outside the scope of this protocol, and we expect that many mechanisms will be used. Once authorized, the presence agent sends a 202 Accepted response. It also sends an immediate NOTIFY message containing the state of the presentity. As the state of the presentity changes, the PA generates NOTIFYs for all

subscribers.

The SUBSCRIBE message effectively establishes a session with the presence agent. As a result, the SUBSCRIBE can be record-routed, and rules for tag handling and Contact processing mirror those for INVITE. Similarly, the NOTIFY message is handled in much the same way a re-INVITE within a call leg is handled.

4 Naming

A presentity is identified in the most general way through a presence URI [4], which is of the form pres:user@domain. These URIs are protocol independent. Through a variety of means, these URIs can be resolved to determine a specific protocol that can be used to access the presentity. Once such a resolution has taken place, the presentity can be addressed with a sip URL of nearly identical form: sip:user@domain. The protocol independent form (the pres: URL) can be thought of as an abstract name, akin to a URN, which is used to identify elements in a presence system. These are resolved to concrete URLs that can be used to directly locate those entities on the network.

When subscribing to a presentity, the subscription can be addressed using the protocol independent form or the sip URL form. In the SIP context, "addressed" refers to the request URI. It is RECOMMENDED that if the entity sending a SUBSCRIBE is capable of resolving the protocol independent form to the SIP form, this resolution is done before sending the request. However, if the entity is incapable of doing this translation, the protocol independent form is used in the request URI. Performing the translation as early as possible means that these requests can be routed by SIP proxies that are not aware of the presence namespace.

The result of this naming scheme is that a SUBSCRIBE request is addressed to a user the exact same way an INVITE request would be addressed. This means that the SIP network will route these messages along the same path an INVITE would travel. One of these entities along the path may act as a PA for the subscription. Typically, this will either be the presence server (which is the proxy/registrar where that user is registered), or the presence client (which is one of the user agents associated with that presentity).

SUBSCRIBE messages also contain logical identifiers that define the originator and recipient of the subscription (the To and From header fields). Since these identifiers are logical ones, it is RECOMMENDED that these use the protocol independent format whenever possible. This also makes it easier to interwork with other systems which recognize these forms.

The Contact, Record-Route and Route fields do not identify logical entities, but rather concrete ones used for SIP messaging. As such, they MUST use the SIP URL forms in both SUBSCRIBE and NOTIFY.

5 Presence Event Package

The SIP event framework [3] defines an abstract SIP extension for subscribing to, and receiving notifications of, events. It leaves the definition of many additional aspects of these events to concrete extensions, also known as event packages. This extension qualifies as an event package. This section fills in the information required by [3].

5.1 Package Name

The name of this package is "presence". This name MUST appear within the Event header in SUBSCRIBE request and NOTIFY request. This section also serves as the IANA registration for the event package "presence".

TODO: Define IANA template in sub-notify and fill it in here.

Example:

Event: presence

5.2 SUBSCRIBE bodies

The body of a SUBSCRIBE request MAY contain a body. The purpose of the body depends on its type. In general, subscriptions will normally not contain bodies. The request URI, which identifies the presentity, combined with the event package name, are sufficient for user presence.

We anticipate that document formats could be defined to act as filters for subscriptions. These filters would indicate certain user presence events that would generate notifies, or restrict the set of data returned in NOTIFY requests. For example, a presence filter might specify that the notifications should only be generated when the status of the users instant message inbox changes. It might also say that the content of these notifications should only contain the IM related information.

5.3 Expiration

User presence changes as a result of events that include:

- Turning on and off of a cell phone
- Modifying the registration from a softphone
- Changing the status on an instant messaging tool

These events are usually triggered by human intervention, and occur with a frequency on the order of minutes or hours. As such, it is subscriptions should have an expiration in the middle of this range, which is roughly one hour. Therefore, the default expiration time for subscriptions within this package is 3600 seconds. As per [3], the subscriber MAY include an alternate expiration time. Whatever the indicated expiration time, the server MAY reduce it but MUST NOT increase it.

5.4 NOTIFY Bodies

The body of the notification contains a presence document. This document describes the user presence of the presentity that was subscribed to. All subscribers MUST support the presence data format described in [fill in with IMPP document TBD], and MUST list its MIME type, [fill in with MIME type] in an Accept header present in the SUBSCRIBE request.

Other presence data formats might be defined in the future. In that case, the subscriptions MAY indicate support for other presence formats. However, they MUST always support and list [fill in with MIME type of IMPP presence document] as an allowed format.

Of course, the notifications generated by the presence agent MUST be in one of the formats specified in the Accept header in the SUBSCRIBE request.

5.5 Processing Requirements at the PA

User presence is highly sensitive information. Because the implications of divulging presence information can be severe, strong requirements are imposed on the PA regarding subscription processing, especially related to authentication and authorization.

A presence agent MUST authenticate all subscription requests. This authentication can be done using any of the mechanisms defined for SIP. It is not considered sufficient for the authentication to be transitive; that is, the authentication SHOULD use an end-to-end mechanism. The SIP basic authentication mechanism MUST NOT be used.

It is RECOMMENDED that any subscriptions that are not authenticated do not cause state to be established in the PA. This can be accomplished by generating a 401 in response to the SUBSCRIBE, and then discarding all state for that transaction. Retransmissions of the SUBSCRIBE generate the same response, guaranteeing reliability even over UDP.

Furthermore, a PA MUST NOT accept a subscription unless authorization has been provided by the presentity. The means by which authorization are provided are outside the scope of this document. Authorization may have been provided ahead of time through access lists, perhaps specified in a web page. Authorization may have been provided by means of uploading of some kind of standardized access control list document. Back end authorization servers, such as a DIAMETER [5], RADIUS [6], or COPS [7], can also be used. It is also useful to be able to query the user for authorization following the receipt of a subscription request for which no authorization information was present. Appendix A provides a possible solution for such a scenario.

The result of the authorization decision by the server will be

reject, accept, or pending. Pending occurs when the server cannot obtain authorization at this time, and may be able to do so at a later time, when the presentity becomes available.

Unfortunately, if the server informs the subscriber that the subscription is pending, this will divulge information about the presentity - namely, that they have not granted authorization and are not available to give it at this time. Therefore, a PA SHOULD generate the same response for both pending and accepted subscriptions. This response SHOULD be a 202 Accepted response.

If the server informs the subscriber that the subscription is rejected, this also divulges information about the presentity - namely, that they have explicitly blocked the subscription previously, or are available at this time and chose to decline the subscription. If the policy of the server is not to divulge this information, the PA MAY respond with a 202 Accepted response even though the subscription is rejected. Alternatively, if the policy of the presentity or the PA is that it is acceptable to inform the subscriber of the rejection, a 603 Decline SHOULD be used.

Note that since the response to a subscription does not contain any useful information about the presentity, privacy and integrity of SUBSCRIBE responses is not deemed important.

5.6 Generation of Notifications

Upon acceptance of a subscription, the PA SHOULD generate an

immediate NOTIFY with the current presence state of the presentity.

If a subscription is received, and is marked as pending or was rejected, the PA SHOULD generate an immediate NOTIFY. This NOTIFY should contain a valid state for the presentity, yet be one which provides no useful information about the presentity. An example of this is to provide an IM URL that is the same form as the presence URL, and mark that IM address as "not available". The reason for this process of "lying" is that without it, a subscriber could tell the difference between a pending subscription and an accepted subscription based on the existence and content of an immediate NOTIFY. The approach defined here ensures that the presence delivered in a NOTIFY generated by a pending or rejected subscription is also a valid one that could have been delivered in a NOTIFY generated by an accepted subscription.

If the policy of the presence server or the presentity is that it is acceptable to divulge information about whether the subscription succeeded or not, the immediate NOTIFY need not be sent for pending or rejected subscriptions.

Of course, once a subscription is accepted, the PA SHOULD generate a NOTIFY for the subscription when it determines that the presence state of the presentity has changed. Section 6 describes how the PA

makes this determination.

For reasons of privacy, it will frequently be necessary to encrypt the contents of the notifications. This can be accomplished using the standard SIP encryption mechanisms. The encryption should be performed using the key of the subscriber as identified in the From field of the SUBSCRIBE. Similarly, integrity of the notifications is important to subscribers. As such, the contents of the notifications SHOULD be authenticated using one of the standardized SIP mechanisms. Since the NOTIFY are generated by the presence server, which may not have access to the key of the user represented by the presentity, it will frequently be the case that the NOTIFY are signed by a third party. It is RECOMMENDED that the signature be by an authority over domain of the presentity. In other words, for a user pres:user@example.com, the signator of the NOTIFY SHOULD be the authority for example.com.

5.7 Rate Limitations on NOTIFY

For reasons of congestion control, it is important that the rate of notifications not become excessive. As a result, it is RECOMMENDED that the PA not generate notifications for a single presentity at a rate faster than once every 5 seconds.

5.8 Refresh Behavior

Since SUBSCRIBE is routed by proxies as any other method, it is possible that a subscription might fork. The result is that it might arrive at multiple devices which are configured to act as a PA for the same presentity. Each of these will respond with a 202 response to the SUBSCRIBE. Based on the forking rules in SIP, only one of these responses is passed to the subscriber. However, the subscriber will receive notifications from each of those PA which accepted the subscriptions. The SIP event framework allows each package to define the handling for this case.

The processing in this case is identical to the way INVITE would be handled. The 202 Accepted to the SUBSCRIBE will result in the installation of subscription state in the subscriber. The subscription is associated with the To and From (both with tags) and Call-ID from the 202. When notifications arrive, those from the PA's whose 202's were discarded in the forking proxy will not match the subscription ID stored at the subscriber (the From tags will differ). These *SHOULD* be responded to with a 481. This will disable the subscriptions from those PA. Furthermore, when refreshing the subscription, the refresh SHOULD make use of the tags from the 202 and make use of any Contact or Record-Route headers in order to deliver the SUBSCRIBE back to the same PA that sent the 202.

The result of this is that a presentity can have multiple PAs active, but these should be homogeneous, so that each can generate the same set of notifications for the presentity. Supporting heterogeneous

PAs, each of which generated notifications for a subset of the presence data, is complex and difficult to manage. If such a feature is needed, it can be accomplished with a B2BUA rather than through a forking proxy.

6 Publication

The user presence for a presentity can be obtained from any number of different ways. Baseline SIP defines a method that is used by all SIP clients - the REGISTER method. This method allows a UA to inform a SIP network of its current communications addresses (ie., Contact addresses) . Furthermore, multiple UA can independently register Contact addresses for the same SIP URL. These Contact addresses can be SIP URLs, or they can be any other valid URL.

Using the register information for presence is straightforward. The address of record in the REGISTER (the To field) identifies the presentity. The Contact headers define communications addresses that describe the state of the presentity. The use of the SIP caller preferences extension [8] is RECOMMENDED for use with UAs that are

interested in presence. It provides additional information about the Contact addresses that can be used to construct a richer presence document. The "description" attribute of the Contact header is explicitly defined here to be used as a free-form field that allows a user to define the status of the presentity at that communications address.

We also allow REGISTER requests to contain presence documents, so that the PUAs can publish more complex information.

Note that we do not provide for locking mechanisms, which would allow a client to lock presence state, fetch it, and update it atomically. We believe that this is not neeeded for the majority of use cases, and introduces substantial complexity. Most presence operations do not require get-before-set, since the SIP register mechanism works in such a way that data can be updated without a get.

The application of registered contacts to presence increases the requirements for authenticity. Therefore, REGISTER requests used by presence user agents SHOULD be authenticated using either SIP authentication mechanisms, or a hop by hop mechanism.

To indicate presence for instant messaging, the UA MAY either register contact addresses that are SIP URLs with the "methods" parameter set to indicate the method MESSAGE, or it MAY register an IM URL.

TODO: This section needs work. Need to define a concrete example of mapping a register to a presence document, once IMPP generates the document format.

6.1 Migrating the PA Function

It is important to realize that the PA function can be colocated with several elements:

- It can be co-located with the proxy server handling registrations for the presentity. In this way, the PA knows the presence of the user through registrations.
- It can be co-located with a PUA for that presentity. In the case of a single PUA per presentity, the PUA knows the state of the presentity by sheer nature of its co-location.
- It can be co-located in any proxy along the call setup path. That proxy can learn the presence state of the presentity by generating its own SUBSCRIBE in order to determine it. In this case, the PA is effectively a B2BUA.

Because of the soft-state nature of the subscriptions, it becomes possible for the PA function to migrate during the lifetime of a subscription. The most workable scenario is for the PA function to migrate from the presence server to the PUA, and back.

Consider a subscription that is installed in a presence server. Assume for the moment that the presence server can determine that a downstream UA is capable of acting as a PA for the presentity. When a subscription refresh arrives, the PA destroys its subscription, and then acts as a proxy for the subscription. The subscription is then routed to the UA, where it can be accepted. The result is that the subscription becomes installed in the PUA.

For this migration to work, the PUA MUST be prepared to accept SUBSCRIBE requests which already contain tags in the To field. Furthermore, the PUA MUST insert a Contact header into the 202, and this header MUST be used by the subscriber to update the contact address for the subscription.

TODO: Does this work? What about getting a Record-Route in place at the PUA. This might only be possible for refreshes that don't use Route or tags.

The presence server determines that a PUA is capable of supporting a PA function through the REGISTER message. Specifically, if a PUA wishes to indicate support for the PA function, it SHOULD include a contact address in its registration with a caller preferences "methods" parameter listing *SUBSCRIBE*.

7 Mapping to CPIM

This section defines how a SIP for presence messages are converted to CPIM, and how a CPIM messages are converted to SIP for presence. SIP to CPIM conversion occurs when a SIP system sends a SUBSCRIBE request that contains a pres URL or SIP URL that corresponds to a user in a domain that runs a different presence protocol. CPIM to SIP involves

the case where a user in a different protocol domain generates a subscription that is destined for a user in a SIP domain.

Note that the process defined below requires that the gateway store subscription state. This unfortunate result is due to the need to remember the Call-ID, CSeq, and Route headers for subscriptions from the SIP side, so that they can be inserted into the SIP NOTIFY generated when a CPIM notification arrives.

7.1 SIP to CPIM

SIP for presnce is converted to CPIM through a SIP to CPIM abstract

gateway service, depicted in Figure 1.

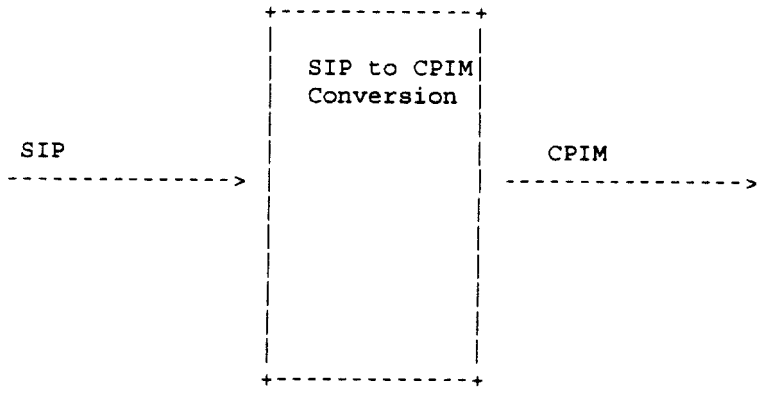

Figure 1: SIP to CPIM Conversion

The first step is that a SUBSCRIBE request is received at a gateway. The gateway generates a CPIM subscription request, with its parameters filled in as follows:

- The watcher identity in the CPIM message is copied from the From field of the SUBSCRIBE. If the From field contains a SIP

URL, it is converted to an equivalent pres URL by dropping all SIP URL parameters, and changing the scheme to pres.

This conversion may not work - what if the SIP URL has no user name. Plus, converting from a URL back to a URN in this fashion may not do it correctly.

- The target identity in the CPIM message is copied from the Request-URI field of the SUBSCRIBE. This may need to be converted to a pres URL as well.

- The duration parameter in the CPIM message is copied from the Expires header in the SUBSCRIBE. If the Expires header specifies an absolute time, it is converted to a delta-time by the gateway. If no Expires header is present, one hour is assumed.

- The transID parameter in the CPIM message is constructed by appending the Call-ID, the URI in the To field, the URI in the From field, the CSeq and the tag in the From field, and the request URI, and computing a hash of the resulting string. This hash is used as the transID. Note that the request URI is included in the hash. This is to differentiate forked requests within the SIP network that may arrive at the same gateway.

The CPIM service then responds with either a success or failure. In the case of success, the SIP to CPIM gateway service generates a 202 response to the SUBSCRIBE. It adds a tag to the To field in the response, which is the same as the transID field in the success response. The 202 response also contains a Contact header, which is the value of the target from the SUBSCRIBE request. It is important that the Contact header be set to the target, since that makes sure that subscription refreshes have the same value in the request URI as the original subscription. The duration value from the CPIM success response is placed into the Expires header of the 202. The gateway stores the Call-ID and Route header set for this subscription.

If the CPIM service responds with a failure, the SIP to CPIM gateway generates a 603 response. It adds a tag to the To field in the response, which is the same as the transID field in the failure response.

When the CPIM system generates a notification request, the SIP to CPIM gateway creates a SIP NOTIFY request. The request is constructed using the standard RFC2543 [2] procedures for constructing a request within a call leg. This will result in the To field containing the watcher field from CPIM, and the From field containing the target field from the CPIM notification. The tag in the From field will

contain the transID. The presence information is copied into the body of the notification. The Call-ID and Route headers are constructed from the subscription state stored in the gateway. If no notification has yet been generated for this subscription, an initial CSeq value

is selected and stored.

SUBSCRIBE refreshes are handled identically to initial subscriptions as above.

If a subscription is received with an Expires of zero, the SIP to CPIM gateway generates an unsubscribe message into the the CPIM system. The watcher parameter is copied from the From field of the SUBSCRIBE. The target parameter is copied from the Request URI field of the SUBSCRIBE. The transID is copied from the tag in the To field of the SUBSCRIBE request.

The response to an unsubscribe is either success or failure. In the case of success, a 202 response is constructed in the same fashion as above for a success response to a CPIM subscriber. All subscription state is removed. In the case of failure, a 603 response is constructed in the same fashion as above, and then subscription state is removed, if present.

7.2 CPIM to SIP

CPIM to SIP conversion occurs when a CPIM subscription request arrives on the CPIM side of the gateway. This scenario is shown in Figure 2.

The CPIM subscription request is converted into a SIP SUBSCRIBE request. To do that, the first step is to determine if the subscribe is for an existing subscription. That is done by taking the target in the CPIM subscription request, and matching it against targets for existing subscriptions. If there are none, it is a new subscription, otherwise, its a refresh.

If its a new subscription, the gateway generates a SIP SUBSCRIBE request in the following manner:

- The From field in the request is set to the watcher field in the CPIM subscription request
- The To field in the request is set to the target field in the CPIM subscription request
- The Expires header in the SUBSCRIBE request is set to the duration field in the CPIM subscription request
- The tag in the From field is set to the transID in the CPIM subscription request.

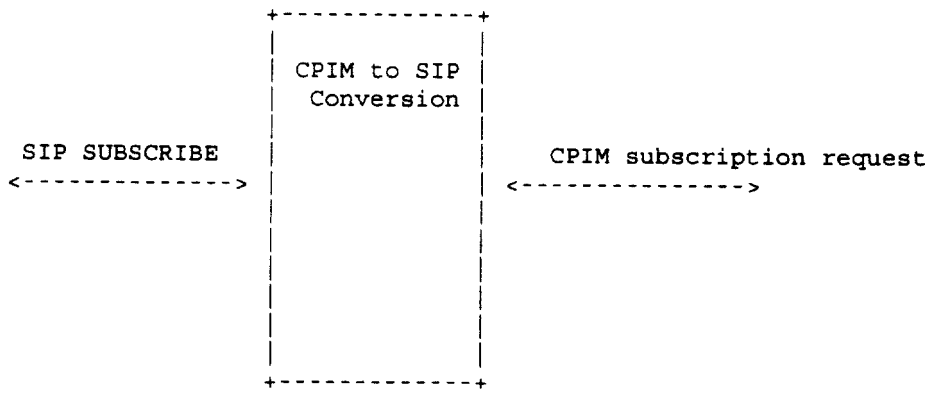

Figure 2: CPIM to SIP Conversion

This SUBSCRIBE message is then sent.

If the subscription is a refresh, a SUBSCRIBE request is generated in the same way. However, there will also be a tag in the To field, copied from the subscription state in the gateway, and a Route header, obtained from the subscription state in the gateway.

When a response to the SUBSCRIBE is received, a response is sent to the CPIM system. The duration parameter in this response is copied from the Expires header in the SUBSCRIBE response (a conversion from an absolute time to delta time may be needed). The transID in the response is copied from the tag in the From field of the response. If the response was 202, the status is set to indeterminate. If the response was any other 200 class response, the status is set to sucess. For any other final response, the status is set to failure.

If the response was a 200 class response, subscription state is