Internet Engineering Task Force
Internet Draft

SIMPLE WG
J. Rosenberg
dynamicsoft
D. Willis
dynamicsoft
H. Schulzrinne
Columbia U.
C. Huitema
Microsoft
B. Aboba
Microsoft
D. Gurle
Microsoft
D. Oran
Cisco

draft-ietf-simple-presence-07.txt
May 20, 2002
Expires: November 2002

Session Initiation Protocol (SIP) Extensions for Presence

STATUS OF THIS MEMO

Abstract

This document describes the usage of the Session Initiation Protocol (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.

Table of Contents

1 Introduction 4
2 Terminology 4
3 Definitions 4
4 Overview of Operation 5
5 Usage of Presence URLs 7
6 Presence Event Package 8
6.1 Package Name 8
6.2 Event Package Parameters 8
6.3 SUBSCRIBE bodies 9
6.4 Subscription Duration 9
6.5 NOTIFY Bodies 9
6.6 Notifier Processing of SUBSCRIBE Requests 10
6.6.1 Authentication 10
6.6.2 Authorization 11
6.7 Notifier Generation of NOTIFY Requests 12
6.8 Subscriber Processing of NOTIFY Requests 13
6.9 Handling of Forked Requests 14
6.10 Rate of Notifications 14
6.11 State Agents 14
7 Publication 16
7.1 Co-location 16
7.2 REGISTER 16
7.3 Uploading Presence Documents 17
8 Example message flow 17
9 Security considerations 20
9.1 Privacy 20
9.2 Message integrity and authenticity 21
9.3 Outbound authentication 21

9.4 Replay prevention 22
9.5 Denial of service attacks 22
9.5.1 Distributed DOS attacks through false contacts 22
10 IANA Consideration 23
11 Contributors 23
12 Acknowledgements 23
13 Authors Addresses 24
14 Normative References 25
15 Informative References 26

1 Introduction

Presence is (indirectly) defined in RFC 2778 [8] 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 the usage of the Session Initiation Protocol (SIP) [1] for presence. This is accomplished through a concrete instantiation of the general event notification framework defined for SIP [2], and as such, makes use of the SUBSCRIBE and NOTIFY methods defined there. Specifically, this document defines an event package, as described in [2]. User presence is particularly well suited for SIP. SIP registrars and location services already hold aspects of 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 event package 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.

This event package is also compliant with the Common Presence and Instant Messaging (CPIM) framework that has been defined in [3]. This allows SIP for presence to easily interwork with other presence systems compliant to CPIM.

2 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [4] and indicate requirement levels for compliant implementations.

3 Definitions

This document uses the terms as defined in RFC 2778 [8]. Additionally, the following terms are defined and/or additionally clarified:

Presence User Agent (PUA): A Presence User Agent manipulates presence information for a presentity. This manipulation can be the side effect of some other action (such as sending a SIP REGISTER request to add a new Contact) or can be done explicitly through the publication of presence documents. We explicitly allow multiple PUAs per presentity. This means that a user can have many devices (such as a cell phone and Personal Digital Assistant (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 knowledge of the presence state of a presentity. This means that it must have access to presence data manipulated by PUAs for the presentity. One way to do this is by co-locating the PA with the

proxy/registrar, or the presence user agent of the presentity. However, this is not the only way, and this specification makes no recommendations about where the PA function should be located. A PA is always addressable with a SIP URI that uniquely identifies the presentity (i.e, sip:joe@example.com). There can be multiple PAs for a particular presentity, each of which handles some subset of the total subscriptions currently active for the presentity. A PA is also a notifier (defined in [2]) that supports the presence events package.

Presence Server: A presence server is a physical 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. When acting as a proxy, the SUBSCRIBE requests are proxied to another entity that may act as a PA.

Edge Presence Server: An edge presence server is a presence agent that is co-located 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.

4 Overview of Operation

In this section, we present an overview of the operation of this event package. The overview describes behavior that is documented in part here, in part within the SIP events framework [2], and in part in the SIP specification [1], in order to provide clarity on this package for readers only casually familiar with those specifications. However, the detailed semantics of this package require the reader to be familiar with SIP events and the SIP specification itself.

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 a SIP URI, SIPS URI [1] or a presence URL [3]. The subscription is carried along SIP proxies as any other request would be. In most cases, 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 an edge presence server. If the edge presence server handles the subscription, it is effectively acting as the presence agent for the presentity. The decision at a presence server 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 edge presence server) 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. If authorized, a 200 OK response is returned. If authorization could not be obtained at this time, the subscription is considered "pending", and a 202 response is returned. In both cases, the PA sends an immediate NOTIFY message containing the state of the presentity and of the subscription. The presentity state may be bogus in the case of a pending subscription, indicating offline no matter what the state of the actual presentity, for example. This is to protect the privacy of the presentity, who may not want to reveal that they have not provided authorization for the subscriber. As the state of the presentity changes, the PA generates NOTIFYs containing those state changes to all subscribers with authorized subscriptions. Changes in the state of the subscription itself can also trigger NOTIFY requests; that state is carried in the Subscription-State header of the NOTIFY, and would typically indicate whether the subscription is active or pending.

The SUBSCRIBE message establishes a "dialog" with the presence agent. A dialog is defined in [1], and it represents the SIP state between a pair of entities to facilitate peer-to-peer message exchanges. This state includes the sequence numbers for messages in both directions (SUBSCRIBE from the subscriber, NOTIFY from the presence agent), in addition to a route set and remote target URI. The route set is a list of SIP (or SIPS) URIs which identify SIP proxy servers that are

to be visited along the path of SUBSCRIBE refreshes or NOTIFY requests. The remote target URI is the SIP or SIPS URI that identifies the target of the message - the subscriber, in the case of NOTIFY, or the presence agent, in the case of a SUBSCRIBE refresh.

SIP provides a procedure called record-routing that allows for proxy servers to request to be on the path of NOTIFY messages and/or

SUBSCRIBE refreshes. This is accomplished by inserting a URI into the Record-Route header in the initial SUBSCRIBE request and/or response.

The subscription persists for a duration that is negotiated as part of the initial SUBSCRIBE. The subscriber will need to refresh the subscription before termination, if they wish to continue. This is accomplished by sending a SUBSCRIBE refresh within the same dialog established by the initial SUBSCRIBE. This SUBSCRIBE is nearly identical to the initial one, but contains the dialog identifier, different sequence numbers, and a set of Route headers that identify the path of proxies the request is to take.

The subscriber can terminate the subscription by sending a SUBSCRIBE, within the dialog, with an Expires header (which indicates duration of the subscription) of zero. This causes an immediate termination of the subscription. A NOTIFY request is then generated by the presence agent with the most recent state. In fact, behavior of the presence agent for handing a SUBSCRIBE with Expires of zero is no different than for any other expiration value; all SUBSCRIBE requests result in a triggered NOTIFY with the current presentity and subscription state.

The presence agent can terminate the subscription at any time. To do so, it sends a NOTIFY request with a Subscription-State header indicating that the subscription has been terminated. A reason parameter can be supplied which provides the reason.

It is also possible to fetch the current presence status, rather than subscribing to it. This is accomplished by sending a SUBSCRIBE request with an immediate expiration.

5 Usage of Presence URLs

A presentity is identified in the most general way through a presence URL [3], which is of the form pres:user@domain. These URLs are protocol independent. They are resolved to protocol specific URIs, such as a SIP or SIPS URI, through DNS procedures defined in [3].

When subscribing to a presentity, the subscription can be addressed using the protocol independent form or the SIP or SIPS URI 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 MAY be used in the Request-URI. In that case, the request would typically be sent to a configured outbound proxy that would perform the resolution. Performing the translation as early as possible means that these requests can be routed by SIP proxies that are not aware of the presence URL.

SUBSCRIBE messages also contain logical identifiers that define the originator and recipient of the subscription (the To and From header fields). These SHOULD contain SIP or SIPS URIs whenever possible, but MAY contain a pres URL if a SIP or SIPS URI is not known or available.

The Contact, Record-Route and Route fields do not identify logical entities, but rather concrete ones used for SIP messaging. SIP [1] specifies rules for their construction.

6 Presence Event Package

The SIP event framework [2] defines a 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 document qualifies as an event package. This section fills in the information required by [2].

6.1 Package Name

The name of this package is "presence". As specified in [2], this value appears in the Event header present in SUBSCRIBE and NOTIFY requests.

Example:

Event: presence

6.2 Event Package Parameters

The SIP Event Framework allows event packages to define additional parameters carried in the Event header for the specific package. This package, presence, does not define any additional parameters.

6.3 SUBSCRIBE bodies

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

We anticipate that document formats could be defined to act as filters for subscriptions. These filters would request that only certain user presence events generate notifies, or ask for a restriction on 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 Instant Message (IM) related information.

Honoring of these filters is at the policy discretion of the PA.

When no body is present, this specifies to the PA that no filter is being requested, so that the PA is being requested to send all NOTIFY requests that its own policy allows.

6.4 Subscription Duration

User presence changes as a result of many events. Some examples are:

- 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 seconds to hours. As such, 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 [2], the subscriber MAY include an alternate expiration time.

6.5 NOTIFY Bodies

As described in [2], the NOTIFY message will contain bodies that

describe the state of the subscribed resource. This body is in a format listed in the Accept header of the SUBSCRIBE, or a package-specific default if the Accept header is omitted.

In this event package, 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 "application/cpim-pidf+xml" presence data format described in [5]. The subscribe request MAY contain an Accept header. If no such header is present, it has a default value of "application/cpim-pidf+xml". If the header is present, it MUST include "application/cpim-pidf+xml", and MAY include any other types capable of representing user presence.

6.6 Notifier Processing of SUBSCRIBE Requests

Based on the proxy routing procedures defined in the SIP specification, the SUBSCRIBE request will arrive at a presence agent (PA). This subsection defines processing at the PA of a SUBSCRIBE request.

If a PA gets a SUBSCRIBE request, and the Request-URI identifies a user the PA is responsible for, but the To header does not, this means that the SUBSCRIBE was forwarded for some reason. Whether the PA is willing to accept subscriptions originally targeted to the user in the To field is a matter of local policy. If a PA decides not to, it SHOULD generate a 403 response.

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.

6.6.1 Authentication

A presence agent MUST authenticate all subscription requests. This authentication can be done using any of the mechanisms defined in [1].

In single-domain systems, where the subscribers all have shared secrets with the PA in the domain, the combination of digest

authentication over Transport Layer Security (TLS) [6] provides a secure and workable solution for authentication. This use case is described in Section 26.3.2.1 of [1].

In inter-domain scenarios, establishing an authenticated identity of the subscriber is harder. It is anticipated that authentication will often be established through transitive trust. Specifically, when user A generates a SUBSCRIBE for B@bar.com, his domain (say, foo.com) will use SIP proxy digest authentication, run over a TLS connection, to identify him (see Section 26.3.2.1 of [1] for an example). The SUBSCRIBE is forwarded to the target domain over a secure connection, such as TLS (see Section 26.3.2.2 of [1] for an example of TLS-based

inter-domain security). The nature of the trust relationship between bar.com and foo.com is that bar.com trusts that foo.com has authenticated all subscribes it receives over that secure connection. As such, the bar.com server need only verify that the SUBSCRIBE came over the secure connection. Any future SIP extensions for network asserted identities could be used in this scenario to allow foo.com to inform bar.com of the authenticated identity.

A presentity MAY choose to represent itself with a SIPS URI. By "represent itself", it means that the user represented by the presentity hands out, on business cards, web pages, and so on, a SIPS URI for their presentity. The semantics associated with this URI, as described in [1], require TLS usage on each hop between the subscriber and the server in the domain of the URI. This provides additional assurances (but no absolute guarantees) that identity has been verified at each hop.

Another mechanism for authentication is S/MIME. Its usage with SIP is described fully in [1]. It provides an end-to-end authentication mechanism that can be used for a PA to establish the identity of the subscriber.

6.6.2 Authorization

Once authenticated, the PA makes an authorization decision. 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 [9], RADIUS [10], or COPS [11], 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. The "watcherinfo" event sub-package for SIP [12] defines a means by which a presentity can become aware that a user has attempted to subscribe to it, so that it can then provide an authorization decision.

Authorization decisions can be very complex. Ultimately, all authorization decisions can be mapped into one of three states: rejected, successful, and pending. Any subscription for which the client is authorized to receive information about some subset of presence state at some points in time is a successful subscription. Any subscription for which the client will never receive any information about any subset of the presence state is a rejected subscription. Any subscription for which it is not known yet whether it is successful or rejected is pending. Generally, pending occurs

when the server cannot obtain authorization at the time of the subscription, and may be able to do so at a later time, when the presentity becomes available.

The appropriate response codes for conveying a successful, rejected, or pending subscription (200, 403 or 603, and 202, respectively) are described in [2].

The SIP events framework allows the initial NOTIFY to contain no body if the resource is not in a meaningful state. In the case of presence, that NOTIFY MAY contain a presence document. This document would indicate whatever presence state the subscriber has been authorized to see; it is interpreted by the subscriber as the current presence state of the presentity. For pending subscriptions, the state of the presentity SHOULD include some kind of textual note that indicates a pending status.

Polite blocking, as described in [13], is possible by generating a 200 OK to the subscription even though it has been rejected (or marked pending). Of course, an immediate NOTIFY will still be sent. The contents of the presence document in such a NOTIFY are at the discretion of the implementor, but SHOULD be constructed in such a

way as to not reveal to the subscriber that their request has actually been blocked. Typically, this is done by indicating "offline" or equivalent status for a single contact address.

6.7 Notifier Generation of NOTIFY Requests

The SIP Events specification details the formatting and structure of NOTIFY messages. However, it leaves to packages the detailed information about what events cause a NOTIFY to be sent, how to compute the state information in the NOTIFY, how to generate neutral or fake state information to hide authorization delays and decisions from users, and whether state information is complete or deltas for notifications.

A PA MAY send a NOTIFY at any time. Typically, it will send ones for successful subscriptions when the state of the presentity changes. The NOTIFY request MAY contain a body indicating the state of the presentity. The times at which the NOTIFY is sent for a particular subscriber, and the contents of the body within that notification, are subject to any rules specified by the authorization policy that governs the subscription. This protocol in no way limits the scope of such policies. As a baseline, a reasonable policy is to generate notifications when the state of any of the communications addresses changes. These notifications would contain the complete and current presence state of the presentity as known to the presence agent. Future extensions can be defined that allow a subscriber to request

that the notifications contain changes in presence information only, rather than complete state.

In the case of a pending subscription, when final authorization is determined, a NOTIFY can be sent. If the result of the authorization decision was success, a NOTIFY SHOULD be sent and SHOULD contain a presence document with the current state of the presentity. If the subscription is rejected, a NOTIFY MAY be sent. As described in [2], the Subscription-State header can indicate the state of the subscription.

The body of the NOTIFY MUST be sent using one of the types listed in the Accept header in the most recent SUBSCRIBE request, or using the type "application/cpim-pidf+xml" if no Accept header was present.

The means by which the PA learns the state of the presentity are also outside the scope of this recommendation. Registrations can provide one way, although the means (if any) by which a PA uses registrations to construct a presence document are an implementation choice. If a PUA wishes to explicitly inform the presence agent of its presence state, it should explicitly upload the presence document (or its piece of it) rather than attempting to manipulate their registrations to achieve the desired result.

For reasons of privacy, it will frequently be necessary to encrypt the contents of the notifications. This can be accomplished using S/MIME. The encryption can 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 MAY provide authentication and message integrity using S/MIME. Since the NOTIFY is 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 is signed by a third party. It is RECOMMENDED that the signature be by an authority over the 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.

6.8 Subscriber Processing of NOTIFY Requests

The SIP Events framework [2] leaves it to event packages to describe the process followed by the subscriber upon receipt of a NOTIFY request, including any logic required to form a coherent resource state.

In this specification, each NOTIFY contains either no presence document, or a document representing the complete and coherent state of the presentity. The presence document in the NOTIFY request with

the highest CSeq value is the current one. When no document is present in that NOTIFY, the presence document present in the NOTIFY with the next highest CSeq value is used. Extensions which specify the use of partial state for presentities will need to dictate how coherent state is achieved.

6.9 Handling of Forked Requests

The SIP Events framework [2] requires each package to describe handling of forked SUBSCRIBE requests.

This specification only allows a single dialog to be constructed as a result of emitting an initial SUBSCRIBE request. This guarantees that only a single PA is generating notifications for a particular subscription to a particular presentity. 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. Doing so would require the subscriber to act as the aggregator for presence data. This aggregation function can only reasonably be performed by agents representing the presentity. Therefore, if aggregation is needed, it MUST be done in a PA representing the presentity that has access to the total set of user presence to be aggregated.

The required processing to guarantee that only a single dialog is established is described in Section 5.4.9 of the SIP Events framework [2].

6.10 Rate of Notifications

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.

6.11 State Agents

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

- It can be co-located with the SIP registrar handling registrations for the presentity (the co-location of the PA within the proxy/registrar is known as a presence server). In this way, the presence server knows the presence of the user through registrations or other means.

- It can be co-located with a PUA for that presentity (the co-

location of the PA within the PUA is known as an edge presence server). 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 server along the request path. That server 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 Back to Back User Agent (B2BUA) for presence. For this mechanism to be effective, all PUA need to act as PA. Therefore, it is RECOMMENDED that all PUA be capable of acting as a PA for the state that they manipulate, and that they authorize subscriptions that can be authenticated as coming from the domain of the presentity.

On occasion, it makes sense for the PA function to migrate from one of these places to another. For example, for reasons of scale, the PA function may reside in the presence server when the PUA is not running, but when the PUA connects to the network, the PA decides to migrate subscriptions to it in order to reduce state in the network. The mechanism for accomplishing the migration is described in Section 4.3.5 of [2]. However, packages need to define under what conditions such a migration would take place.

A PA MAY choose to migrate subscriptions at any time, through configuration, or through dynamic means. One dynamic means for a presence server to discover that the function can migrate to a PUA is 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]. This indicates that it is capable of terminating and processing SUBSCRIBE, and therefore may be able to act as a PA. However, just because a PUA indicates it can accept subscriptions, does not mean a PA should migrate the subscriptions there. In particular, a PA SHOULD NOT migrate the subscription if it is composing aggregated presence documents from state received from several PUA.

When the PA sends notifications to migrate subscriptions, it should be wary of the load that this may cause. A PA SHOULD rate limit the notifications, in order to avoid a flood of simultaneous re-SUBSCRIBEs from all subscribers.

In the case where the subscription has migrated to the presence server, the presence server will simply act as a PA for these new subscriptions. In the case where the subscription has migrated from the presence server to the PUA, the presence server MUST operate like

a proxy. Furthermore, it SHOULD implement the SIP Caller preferences extension [7]. Because of the existence of a registered Contact with a "methods" parameter containing SUBSCRIBE, the caller preferences extension will cause the proxy to send the SUBSCRIBEs to that Contact. Assuming it accepts, a 2xx is generated and forwarded to the subscriber. The subscriber will now receive and accept notifications from that PA. Because the "methods" parameter does not convey the set of event packages for which the PUA can accept SUBSCRIBE, it is possible that the PUA doesn't understand the presence event package, and will therefore reject the subscription with a 489. In this case, the presence server SHOULD act as a PA for this subscription and generate its own response. Furthermore, it SHOULD NOT migrate any other subscriptions to this PUA.

Migration of subscriptions will still work if the proxy does not support the caller preferences extension. However, the proxy will instead fork the SUBSCRIBE, possibly to Contacts which have not indicated that they support SUBSCRIBE. The result will be 405 responses from those UAS. However, the one UAS which does support the method will generate a 2xx class response (assuming the subscription is accepted), and this will be correctly forwarded towards the subscriber based on proxy response processing rules [1]. The penalty of not supporting caller preferences is the additional unneeded SIP traffic.

7 Publication

The user presence for a presentity can be obtained from any number of different ways. None of these mechanisms are mandated by this specification. The discussion here is for informational purposes only.

7.1 Co-location

When the PA function is co-located with the PUA, user presence is known directly by the PA.

7.2 REGISTER

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.

Usage of REGISTER information to construct presence is only possible if the PA is co-located with, or shares information with, the SIP

registrar. In this case, the combined PA/registrar/proxy is known as a presence server.

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 [7] 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 presence of a registered Contact with a "methods" parameter [7] listing the MESSAGE method implies that the presentity supports instant messaging as a communications means.

The q values from the Contact header [1] can be used to establish priorities amongst the various communications addresses in the Contact headers.

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

7.3 Uploading Presence Documents

If a means exists to upload presence documents from PUA to the PA, the PA can act as an aggregator and redistributor of those documents. The PA, in this case, would take the presence documents received from each PUA for the same presentity, and merge the communications means across all of those PUA into a single presence document. Typically, this aggregation would be accomplished through administrator or user defined policies about how the aggregation should be done.

The specific means by which a presence document are uploaded to a presence agent are outside the scope of this specification. When a PUA wishes to have direct manipulation of the presence that is distributed to subscribers, direct uploading of presence documents is

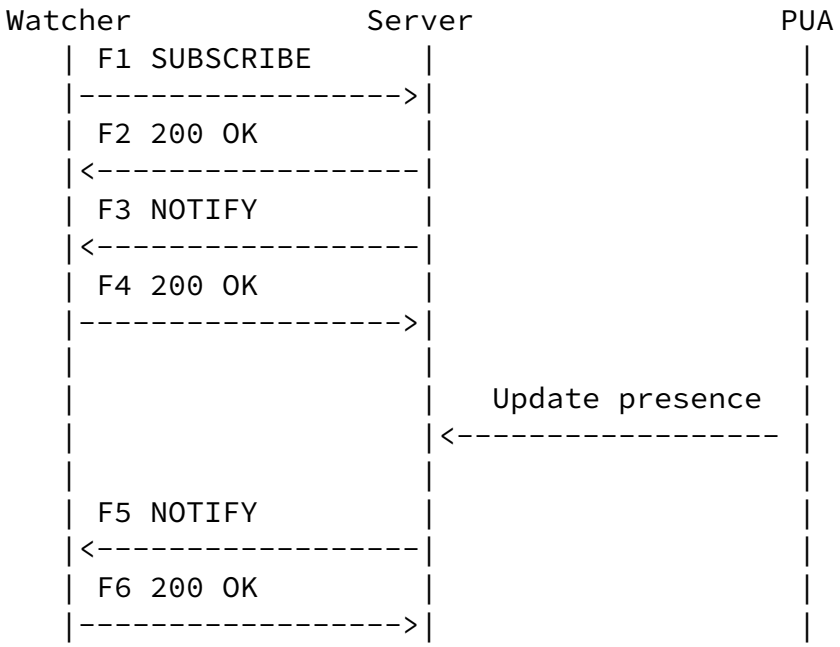
RECOMMENDED.

8 Example message flow

This message flow illustrates how the presence server can be the responsible for sending notifications for a presentity. This flow assumes that the watcher has previously been authorized to subscribe to this resource at the server.

In this flow, the PUA informs the server about the updated presence information though some non-SIP means.

When the value of the Content-Length header is "..." this means that the value should be whatever the computed length of the body is.



Message Details

F1 SUBSCRIBE watcher->example.com server

```
SUBSCRIBE sip:resource@example.com SIP/2.0
Via: SIP/2.0/UDP watcherhost.example.com;branch=z9hG4bKnashds7
To: <sip:resource@example.com>
From: <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Max-Forwards: 70
Event: presence
Accept: application/cpim-pidf+xml
Contact: <sip:user@watcherhost.example.com>
Expires: 600
Content-Length: 0
```


F2 200 OK example.com server->watcher

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP watcherhost.example.com;branch=z9hG4bKnashds7
  ;received=192.0.2.1
To: <sip:resource@example.com>;tag=ffd2
From: <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Event: presence
Expires: 600
Contact: sip:server.example.com
Content-Length: 0
```

F3 NOTIFY example.com server-> watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/UDP server.example.com;branch=z9hG4bKna998sk
From: <sip:resource@example.com>;tag=ffd2
To: <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
```

```
   Event: presence
   Subscription-State: active;expires=599
   Max-Forwards: 70
   CSeq: 8775 NOTIFY
   Content-Type: application/cpim-pidf+xml
   Content-Length: ..

   [PIDF Document]
```

F4 200 OK watcher-> example.com server

```
   SIP/2.0 200 OK
   Via: SIP/2.0/UDP server.example.com;branch=z9hG4bKna998sk
     ;received=192.0.2.2
   From: <sip:resource@example.com>;tag=ffd2
   To: <sip:user@example.com>;tag=xfg9
   Call-ID: 2010@watcherhost.example.com
   CSeq: 8775 NOTIFY
```



```
   Content-Length: 0
```

F5 NOTIFY example.com server -> watcher

```
   NOTIFY sip:user@watcherhost.example.com SIP/2.0
   Via: SIP/2.0/UDP server.example.com;branch=z9hG4bKna998sl
   From: <sip:resource@example.com>;tag=ffd2
   To: <sip:user@example.com>;tag=xfg9
   Call-ID: 2010@watcherhost.example.com
   CSeq: 8776 NOTIFY
   Event: presence
   Subscription-State: active;expires=543
   Max-Forwards: 70
   Content-Type: application/cpim-pidf+xml
```

```
   Content-Length: ...

   [New PIDF Document]
```

```
F6 200 OK

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP server.example.com;branch=z9hG4bKna998sl
    ;received=192.0.2.2
   From: <sip:resource@example.com>;tag=ffd2
   To: <sip:user@example.com>;tag=xfg9
   Call-ID: 2010@watcherhost.example.com
   CSeq: 8776 NOTIFY
   Content-Length: 0
```

9 Security considerations

There are numerous security considerations for presence. Many are outlined above; this section considers them issue by issue.

9.1 Privacy

Privacy encompasses many aspects of a presence system:

- Subscribers may not want to reveal the fact that they have subscribed to certain users
- Users may not want to reveal that they have accepted subscriptions from certain users
- Notifications (and fetch results) may contain sensitive data which should not be revealed to anyone but the subscriber

Privacy is provided through a combination of hop-by-hop encryption and end-to-end encryption. The hop-by-hop mechanisms provide scalable privacy services, disable attacks involving traffic analysis, and hide all aspects of presence messages. However, they operate based on transitivity of trust, and they cause message content to be revealed to proxies. The end-to-end mechanisms do not require transitivity of trust, and reveal information only to the desired recipient. However, end-to-end encryption cannot hide all information, and is susceptible to traffic analysis. Strong end to end authentication and encryption also requires that both participants have public keys, which is not generally the case. Thus, both mechanisms combined are needed for complete privacy services.

SIP allows any hop by hop encryption scheme, but TLS is mandatory to implement for servers. Therefore, it is RECOMMENDED that TLS [6] be used between elements to provide this function. The details for usage of TLS for server-to-server, and client-to-server security are detailed in Section 26.3.2 of SIP [1].

SIP encryption, using S/MIME, MAY be used end-to-end for the transmission of both SUBSCRIBE and NOTIFY requests.

9.2 Message integrity and authenticity

It is important for the message recipient to ensure that the message contents are actually what was sent by the originator, and that the recipient of the message be able to determine who the originator really is. This applies to both requests and responses of SUBSCRIBE and NOTIFY. This is supported in SIP through end-to-end authentication and message integrity. SIP provides http digest for authentication, and S/MIME for authentication and integrity.

9.3 Outbound authentication

When local proxies are used for transmission of outbound messages, proxy authentication is RECOMMENDED. This is useful to verify the identity of the originator, and prevent spoofing and spamming at the

originating network.

9.4 Replay prevention

To prevent the replay of old subscriptions and notifications, all signed SUBSCRIBE and NOTIFY requests and responses MUST contain a Date header covered by the message signature. Any message with a date older than several minutes in the past, or more than several minutes into the future, SHOULD be discarded.

Furthermore, all signed SUBSCRIBE and NOTIFY requests MUST contain a Call-ID and CSeq header covered by the message signature. A user agent or presence server MAY store a list of Call-ID values, and for each, the highest CSeq seen within that Call-ID. Any message that arrives for a Call-ID that exists, whose CSeq is lower than the highest seen so far, is discarded.

Finally, HTTP digest authentication MAY be used to prevent replay attacks.

9.5 Denial of service attacks

Denial of Service (DOS) attacks are a critical problem for an open, inter-domain, presence protocol. Here, we discuss several possible attacks, and the steps we have taken to prevent them.

9.5.1 Distributed DOS attacks through false contacts

Unfortunately, presence is a good candidate for Distributed DOS (DDOS) attacks because of its amplification properties. A single SUBSCRIBE message could generate a nearly unending stream of notifications, so long as a suitably dynamic source of presence data can be found. Thus, a simple way to launch an attack is to send subscriptions to a large number of users, and in the Contact header (which is where notifications are sent), place the address of the target.

The only reliable way to prevent these attacks is through authentication and authorization. End users will hopefully not accept subscriptions from random unrecognized users. Also, the presence client software could be programmed to warn the user when the Contact header in a SUBSCRIBE is from a domain which does not match that of the From field (which identifies the subscriber).

Also, note that as described in [2], if a NOTIFY is not acknowledged or was not wanted, the subscription that generated it is removed. This eliminates the amplification properties of providing false Contact addresses.

10 IANA Consideration

This specification registers an event package, based on the registration procedures defined in [2]. The following is the information required for such a registration:

Package Name: presence

Package or Template-Package: This is a package.

Published Document: RFC XXXX (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification).

Person to Contact: Jonathan Rosenberg, jdrosen@jdrosen.net.

11 Contributors

The following individuals were part of the initial team that worked through the technical design of this specification:

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

Robert Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
email: rsparks@dynamicsoft.com

Ben Campbell
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
email: bcampbell@dynamicsoft.com

12 Acknowledgements

We would like to thank Rick Workman, Adam Roach, Sean Olson, Billy

Biggs, Stuart Barkley, Mauricio Arango, Richard Shockey, Jorgen

Bjorkner, Henry Sinnreich, Ronald Akers, Paul Kyzivat, and Hisham Khitarbil for their comments and support of this specification.

13 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Dean Willis
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
email: dwillis@dynamicsoft.com

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

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

14 Normative References

[1] J. Rosenberg, H. Schulzrinne, et al. , "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[2] A. Roach, "SIP-specific event notification," Internet Draft, Internet Engineering Task Force, Mar. 2002. Work in progress.

[3] D. Crocker et al. , "Common presence and instant messaging (CPIM)," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

---

[4] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[5] H. Sugano, S. Fujimoto, et al. , "CPIM presence information data format," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[6] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, Internet Engineering Task Force, Jan. 1999.

[7] H. Schulzrinne and J. Rosenberg, "SIP caller preferences and callee capabilities," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

15 Informative References

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

[9] J. Arkko, P. Calhoun, et al. , "Diameter base protocol," Internet Draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[10] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote authentication dial in user service (RADIUS)," RFC 2865, Internet

Engineering Task Force, June 2000.

[11] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry, "The COPS (common open policy service) protocol," RFC 2748, Internet Engineering Task Force, Jan. 2000.

[12] J. Rosenberg, "A SIP event sub-package for watcher information," Internet Draft, Internet Engineering Task Force, Mar. 2002. Work in progress.

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