SIMPLE
Internet-Draft
Expires: August 12, 2003

B. Campbell
dynamicsoft
S. Olson
Microsoft
J. Peterson
NeuStar, Inc.
J. Rosenberg
dynamicsoft
B. Stucker
Nortel Networks, Inc.
February 11, 2003

# SIMPLE Presence Publication Requirements
# draft-ietf-simple-publish-reqs-00

## Status of this Memo



## Copyright Notice



## Abstract

This document describes requirements for an extension to the Session Initiation Protocol (SIP) [1]. The purpose of this extension would be to create a means for publishing event state used within the framework for SIP Event Notification (RFC3265 [2]). The first

application of this extension is targeted at the publication of presence information as defined by the SIMPLE [5] working group.

# 1. Introduction

## 1.1 Publication Model

The following sections outline a model for publication of event state, in particular presence information. This model further defines the problem that this mechanism is attempting to solve.

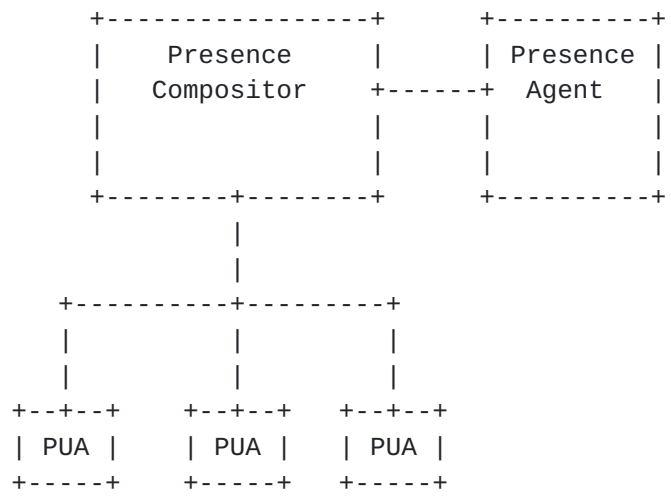
The method described in this document allows presence information to be published to a presence agent on behalf of a user. This method can be extended to support publication of other event state, but it is not intended to be a general-purpose mechanism for transport of arbitrary data as there are better suited mechanisms for this purpose (ftp, http, etc.) This method is intended to be a simple, light-weight mechanism that employs SIP in order to support SIMPLE services.

### 1.1.1 Presence Composition

Most existing presence services involve a single PUA that has complete presence for a given presentity. This allows for a very simple model where that PUA sends full presence state to a PA, which then distributes it to watchers.

But this is a limited view of presence. In general, the presence state of a presentity may be derived from many different inputs. A complete view of presence for a presentity is likely to be derived from more than one source, where the the complete view of presence state is composed of the presence state from each source. Therefore, any given PUA is unlikely to be able to provide complete presence information. This document proposes a logical model for publishing presence information from multiple PUAs to a presence compositor, that can act as a presence agent.

Presence composition is a logical function in a presence distribution system. This function is fulfilled by a logical element known as a "presence compositor". A presence compositor accepts presence inputs from one or more PUAs, and composes these inputs into a composite presence document.




Each PUA publishes its view of presence to the Presence Compositor, which then relays the composed presence information to a presence agent for distribution. It must be possible for each PUA publishes a full view of presence from its perspective--each publication carries full state, and does not depend on previous states for the particular PUA. A PUA does not need to know whether or not other PUAs are also publishing presence information to the compositor.

The transformations that a presence compositor uses for this composition are entirely a matter of local policy. The policy could be as simple as the creation of a combined CPIM PIDF [4] document where each input represents a separate tuple. It can also involve more complex transformations, such as modifying the information from one source based on information from another source.

### 1.1.2 Interface between the Compositor and the PA

The interface between a compositor and a PA is also a matter of local policy. The compositor might act as a PUA itself, and publish presence to the PA just like any other PUA might. The compositor and the PA may be co-located, and communicate via local procedure calls. Specification of this interface is beyond the scope of this document.

## 1.2 Why use SIP for publication?

Two principal protocol candidates have been evaluated for presence state publication from a PUA to a presence compositor: HTTP and SIP. This document recommends the use of SIP for presence publication for the following reasons.

### 1.2.1 HTTP

HTTP is well suited for moving data around in the form of MIME body parts. An HTTP client-to-server publication solution would not require much work to specify. A SOAP over HTTP solution would additionally allow complex transaction semantics with little additional work. HTTP, however, does not have a well-defined routing model at the application level. It works fine if the publication point is well known and fairly static, but it will require additional work to deal with situations where the publication point changes dynamically.

### 1.2.2 SIP

SIP, on the otherhand, is built around the concept of mobility of endpoints. The SIP proxy, registrar, and location service concepts provide a rich mechanism of finding a dynamically changing endpoint from and address of record. The application-level routing capabilities of SIP can be very useful for presence publication. If all PUAs for a given presentity exist in the same administrative domain, then they can most likely publish directly to a compositor. But if PUAs exist outside the administrative domain, it is likely they will not be able to do so.

For example, suppose that Alice uses a presence service that allows multiple PUAs to publish to a compositor inside the service provider network. Further suppose that Alice wishes to incorporate presence information from an external provider, that has no business relationship with her primary provider. For this example, Alice wishes to use a shared browsing service that tracks the "location" she is currently browsing in the web. That service acts as a PUA on Alice's behalf, and publishes the information to her primary presence provider. Other users of the shared browsing service can subscribe to her presence information, and determine when they are browsing the same site.

The presence provider is highly unlikely to allow the external PUA to send data directly to the compositor. But if Alice registers a contact with a methods parameter value of "PUBLISH", that PUA can send a publish request to an edge proxy in the presence providers network, and use Alice's address of record as the requestURI. This AoR could be her normal SIP URI, or it could be a special AoR for the purposes of presence publication. The proxy forwards the request to the compositor, without the external PUA having to talk directly to the compositor, or even know its IP address.

Now consider that Alice's primary providor is actually an enterprise. That enterprise has different compositors for different departments.

The external provider has no way of knowing this internal organization, nor does it know what department Alice works in. Still, Alice register's her publication contact at an enterprise registrar, the external provider sends the publish request to Alice's address of record, and the companies internal SIP network handles things from there, eventually getting the request to the correct compositor.

The composition model does not at first appear to require publishing to dynamically changing PAs. But a very powerful, but often forgotten, aspect of SIMPLE is it allows a PA to exist on an end-user device. Indeed, some early implentations of SIMPLE rely on exactly that model. It is reasonable to expect the composition model above to co-exist with end-user device PAs, where the PA location changes dynamically.

For example, imagine Alice hosts a PA on her PC, which aquires its IP address via DHCP. This address changes relatively frequently, and registers a publish contact with an enterprise registrar. Now, imagine she also has a mobile phone which contains a PUA. She wants her presence document to show a combined view of her PCs concept of her presence and her mobile phone service's concept of her precence. She cannot simply tell the mobile service her PC address since it changes often. Instead, she tells the service an AoR to publish presence to. The mobile service publishes presence state to that AoR, which resolves to a SIP proxy or redirect server. Normal SIP proxy or redirect behavior is invoked to get the publication to Alice's PC based on her publish contact registration.

### 1.2.3 Conclusions

It is our opinion that SIP style routing is very useful for presence publication. Without the application level routing capabilities of SIP, it would be difficult to build these sort of services. It is more appropriate to add a publication mechanism to SIP than to standardize SIP-style routing features for HTTP proxies.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3].

## 3. Publication Requirements

Any presence publication mechanism for SIP must meet the following requirements.

1. The presence publication mechanism must provide a way for a PUA to publish presence to a presence compositor.

2. The presence publication mechanism must define a new SIP method for presence publication, or describe how some existing method can perform this function.

3. The presence publication mechanism must define a mandatory-to-implement format for expressing presence information. This format must have a registered MIME type. It is recommended that this format be the Presence Information Data Format [4]. Other formats in addition to the single mandatory-to-implement format may be suggested.

4. The publication mechanism must support the simultaneous publication from several distinct PUAs of presence information concerning the same presentity.

5. The publication mechanism must allow for the fact that any individual PUA may not know the total presence state of the user. Publications from any given PUA may contain only part of the total presence of the presentity.

6. It must be possible to order a published sequence of presence information originating from a particular PUA.

7. It must be possible for a compositor to reject a publication request.

8. The publication mechanism must support a means for a presence compositor to compose presence information received from multiple PUAs at different times into a single presence document. The compositor function must be able to detect and reconcile conflicts or overlaps in publications from different PUAs.

9. The publication mechanism must support a means for large content to be sent as a part of presence information.

10. It must be possible for a PUA to send full presence information (all of the presence information that the PUA knows about a presentity) in a publication, and there should also be a way for PUAs to send partial or incremental presence information.

11. The publication mechanism must support a way to uniquely identify segments of presence information (for example, the tuple elements in PIDF are uniquely identified by a tuple ID).

12. There must be a way to uniquely identify segments of presence information for a particular presentity that is independent of the PUA that generated this segment. The tuple identifier from PIDF is an example of such a unique identifier.

13. The publication mechanism must allow two PUAs to publish information for the same segment of presence information.

14. PUAs must have a capability that allows them to query for the identifiers of all of the segments of presence information that have currently been published for a presentity (provided that the PUA is authorized to receive this information).

15. It must be possible for the publication of presence information for a particular segment to overwrite existing information for that segment, even if the existing information had been published by a different PUA. Overwrites could occur, for example, on the basis of a timestamp indicating when the segment was last modified (i.e. more recent segments overwrite older segments).

16. There must be a way for a compositor to reject a published segment of presence information from a PUA because the segment has been replaced by one published by another PUA.

17. It must be possible for a compositor to authenticate a publisher of presence information.

18. The presence publication architecture must support a means for principals to authorize the disclosure of segments of presence information to particular watchers.

19. There must be a way for a publisher to tell a presence agent that a piece of published presence should be passed on to watchers without modification.

## 4. IANA Considerations

This document introduces no considerations for IANA.

## 5. Security Considerations

A presence compositor should authenticate publishing User Agents, and may apply authorization policies. The composition model makes no assumptions that all input sources for a compositor are on the same network, or in the same administrative domain.

Authorization of watchers becomes more complicated in a presence publication architecture. Principals should have some way to manage the authorization of watchers who wish to published presence information. Since they may not directly control the presence agent to which watchers subscribe, this introduces some mechanism requirements.

The compositor should throttle incoming publications and the corresponding notifications resulting from the changes in event state. As a first step careful selection of default Expires: values for the supported event packages at a compositor can help limit refreshes of event state. Additional throttling and debounce logic at the compositor is advisable to further reduce the notification traffic produced as a result of a PUBLISH method.

## 6. Acknowledgments

The authors would like to thank Krisztian Kiss and Aki Niemi for their suggestions.

Normative References

[1] Rosenberg, J., Schulzrinne, Camarillo, Johnston, Peterson, Sparks, Handley and Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Roach, A., "Session Initiation Protocol(SIP)-Specific Event Notification", RFC 3265, June 2002.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[4] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W. and J. Peterson, "Common Presence and Instant Messaging (CPIM) Presence Information Data Format", draft-ietf-impp-cpim-pidf-07 (work in progress), May 2002.

[5] <http://www.ietf.org/html.charters/simple-charter.html>

## Authors' Addresses

Ben Campbell
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75025
US

EMail: bcampbell@dynamicsoft.com

Sean Olson
Microsoft
One Microsoft Way
Redmond, WA 98052
US

Phone: +1-425-707-2846
EMail: seanol@microsoft.com
URI: http://www.microsoft.com/rtc

Jon Peterson
NeuStar, Inc.
1800 Sutter St
Suite 570
Concord, CA 94520
US

Phone: +1-925-363-8720
EMail: jon.peterson@neustar.biz
URI: http://www.neustar.biz

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
US

EMail: jdrosen@dynamicsoft.com

Brian Stucker
Nortel Networks, Inc.
2380 Performance Drive
Richardson, TX 75082
US

EMail: bstucker@nortelnetworks.com

## Full Copyright Statement



## Acknowledgement

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