SIMPLE
Internet-Draft
Expires: August 28, 2003

B. Campbell
dynamicsoft
S. Olson
Microsoft
J. Peterson
NeuStar, Inc.
J. Rosenberg
dynamicsoft
B. Stucker
Nortel Networks, Inc.
February 27, 2003

SIMPLE Presence Publication Mechanism
draft-olson-simple-publish-02

Status of this Memo

Copyright Notice

Abstract

This document describes an extension to the Session Initiation Protocol (SIP) for publishing event state used within the framework for SIP Event Notification. The first application of this extension is targeted at the publication of presence information.

The method described in this document allows event 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.

Table of Contents

1. Introduction

The focus of this specification is to provide a framework for the publication of event state from a UA to a entity that is responsible for compositing this event state and distributing that state to interested parties through the SUBSCRIBE/NOTIFY mechanism. This specification fills a current gap in the event notification framework to allow for a client to push its state to the state agent that acts on its behalf. It is the intention of this framework to allow any event state for which there is an appropriate event package (as defined in RFC 3265 [2]) to be published.

The first application of this mechanism is the publication of presence state by a PUA to a presence compositor which has a tightly coupled relationship to the PA. The requirements and model for presence publication are documented in [4]. This specification will address each of these requirments.

To accomplish this task a new SIP method, PUBLISH, is defined. PUBLISH is analogous to REGISTER in that it allows a UA to add, modify, and remove state in another entity which manages this state on behalf of a user. The user may in turn have multiple UAs or endpoints. Each endpoint may publish its own unique state and through a subscription to that event discover the event state of the other endpoints for a user. PUBLISH is defined to create soft-state in the state agent; this state has a defined lifetime and will expire after a negotiated amount of time. Local policy at the compositor may in turn define hard-state for this event package. That is, the steady-state of this event package in the absence of any other soft-state provided through the PUBLISH method. In the generic sense, a UAC which publishes event state is labelled an Event Publication Agent (EPA). For presence in particular, this is the familiar PUA role as defined in [7]. The entity which processes the PUBLISH request is known as a Event State Compositor (ESC). For presence in particular, this is the familiar PA role.

Event state publication inherently involves at least two parties: the source of the publication and the target of the publication. The source of the publication is naturally represented as an address-of-record (AOR). For some types of event state, namely presence, the target of the publication may not sufficiently be represented by an address-of-record (AOR) alone. Rather, the target is a combination of both an AOR and a unique identifier which acts to represent one of N possible sections of an overall event state for that AOR. In this specification, these sections are referred to as event state segments. In the context of presence publication, the event state segment is nothing more than the presence tuple associated with the presentity (AOR). It is the role of the

compositor to aggregate these segments into a complete event state which is presented to the subscribers of that event state. This composition logic is a matter of local policy. For some event packages, there is no natural decomposition of event state into these segments and for these packages, an AOR is sufficient to identify the target of the publish.

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].

In addition to the terminology of RFC 3265 [2], this document introduces some new concepts:

Event Publication Agent (EPA): The UAC which issues a PUBLISH request to publish event state.

Event State Compositor (ESC): The UAS which processes PUBLISH requests and composites the event state. It is assumed that there is a tight coupling between the ESC which receives the PUBLISH requests and the state agent which issues appropriate NOTIFY requests based on this change in event state. The interface between these two components is out-of-scope for this specification.

Event State Segment: The EPA may publish event state that is divided into individual segments. For the presence publication case, these segments are the individual tuples in the presence document. In the generic case, this document will refer to these segments as event state segments.

3. Constructing the PUBLISH Request

PUBLISH requests create, remove, and modify event state. A PUBLISH request can create new event state in the state agent, associating this event state with an address-of-record and optionally with a unique identifier for segments of event state being published. Publication on behalf of a particular address-of-record may also be performed by a suitably authorized third party. To determine the current published state for a particular address-of-record, the client MAY create a subscription for this address-of-record and event package using the SUBSCRIBE/NOTIFY mechanism of RFC3265.

Note that in the case the event state is segmented, each segment logically represents an independent publication that may be added, removed, modified, and expired separately. For presence publication, this means each tuple in the PIDF document in the PUBLISH request is logically a separate publication that may be manipulated independently even though they are grouped together in the same PIDF document initially.

Except as noted, the construction of the PUBLISH request and the behavior of clients sending a PUBLISH request is identical to the general UAC behavior described in Section 8.1 and Section 17.1 of RFC 3261 [1].

A PUBLISH request does not establish a dialog. A UAC MAY include a Route header field in a PUBLISH request based on a pre-existing route set as described in Section 8.1 of RFC3261. The Record-Route header field has no meaning in PUBLISH requests or responses, and MUST be ignored if present. In particular, the UAC MUST NOT create a new route set based on the presence or absence of a Record-Route header field in any response to a PUBLISH request. The PUBLISH request MUST NOT contain a Contact header.

The following header fields MUST be included in a PUBLISH request:

Request-URI: The Request-URI initially contains the address-of-record whose publication is to be created, removed, or modified. Unlike the REGISTER request, the Request-URI SHOULD contain a SIP(S) URI with a username. The address-of-record MUST be a SIP URI or SIPS URI.

To: The To header field contains the address of record whose publication is to be created, removed, or modified. The To header field and the Request-URI field are typically the same. This address-of-record MUST be a SIP URI or SIPS URI.

From: The From header field contains the address-of-record of the person responsible for the publication. The value is the same as the To header field unless the request is a third- party publication.

Call-ID: All publications from an EPA MAY use the same Call-ID header field value for publications sent to a particular state agent.

CSeq: An EPA MUST increment the CSeq value by one for each PUBLISH request with the same Call-ID. Unlike REGISTER requests, the Call-ID and CSeq are not directly used for ordering of PUBLISH requests.

Event: PUBLISH requests MUST contain a single Event header field. This value indicates the event package which this request is publishing state for.

Expires: PUBLISH requests SHOULD contain a single Expires header field. This value indicates the lifetime of the event state being published by this request. A special value of "0" indicates the removal of any prior soft-state established by a prior PUBLISH request from this UAC.

The body of the PUBLISH request contains the event state that the client wishes to publish. The content format and semantics are dependent on the event package identified in the Event header. Any event package which makes use of the PUBLISH mechanism MUST describe these semantics and MUST prescribe a default, mandatory to implement format. This document defines the semantics of the presence publication requests (event package "presence") when the CPIM PIDF [5] presence document format is used.

4. Requirements of the body of a PUBLISH request

In order to satisfy the requirements of [4], the body of the PUBLISH request must fulfill several requirements as well. Any application of the PUBLISH mechanism for a given event package MUST support a Content-Type which fulfills these requirements. For presence publication, it will be demonstrated how these requirements may be fulfilled using the CPIM PIDF presence format in [5] within a PUBLISH request. A PUA which uses PUBLISH to publish presence state to the PA MUST support the CPIM PIDF presence format.

The content type MUST provide a way to indicate an ordering of publication requests. For example, the timestamp element in PIDF provides a temporal ordering of presence state changes that allows the Event State Compositor (ESC) to properly order PUBLISH requests. When used as the content of a PUBLISH request, the PUA MUST supply a timestamp element for every presence tuple present in the PIDF document.

The content type MUST provide a way to publish partial state for an event package. The intention is to allow each device or client for an address-of-record to publish event state independently. To accomplish this, the event state that is published by these devices must be allowed to be only a portion of the complete state that the state agent advertises for that AOR. For example, a PUA can publish presence state for just a subset of the tuples that may be composited into the presence document that watchers receive in a NOTIFY. The mechanism by which the ESC aggregates this information is a matter of local policy.

If the content type allows for event state segments to be represented, the content type MUST provide a means to uniquely identify each unique segment. For example, the CPIM PIDF presence document provides a tuple ID to distinguish the segments of the presence document associated with the encompassing presentity.

As with any other SIP message, the PUBLISH mechanism MAY use the content indirection mechanism defined in [6]. There are no additional requirements or restrictions on content indirection as applied to the PUBLISH request. Content indirection is a useful mechanism for communicating large event state information that cannot be carried directly within the SIP signaling (PUBLISH request).

5. Creating Initial Publication Soft-State

The PUBLISH request created by the EPA and sent to the Event State Compositor (ESC) establishes soft-state in the state agent for the event package indicated in the request and bound to the address-of-record in the To header of the request. Additionally, the PUBLISH request may publish event state that is further sub-divided into segments of event state that may be manipulated independently. As an example, presence publication using the CPIM PIDF format may manipulate individual tuples related to a common presentity.

Once the initial PUBLISH request has been processed by the ESC, the EPA MAY send subsequent PUBLISH requests to refresh, modify, or delete the publication state established by the first PUBLISH request. These operations will be described in subsequent sections.

EPAs MUST NOT send a new PUBLISH request (not a re-transmission) until they have received a final response from the state agent for the previous one or the previous PUBLISH request has timed out.

6. Setting the Expiration Interval of Event State

When a client sends a PUBLISH request, it SHOULD suggest an expiration interval that indicates how long the client would like the publication to be valid. The actual duration of the soft state is defined by local policy at the ESC. The expiration value is presented in the Expires header of the PUBLISH request. If an Expires header is not present, the client is indicating its desire for the server to choose. It is RECOMMENDED that the PA use a value of 3600 seconds (1 hour) for this default expiration value in the case of presence publication. The default value is generally event package specific.

7. Removing Event State

PUBLISH establishes soft state which expires unless refreshed. This event state may also be explicitly removed. A UA requests the immediate removal of event state by specifying an Expires value of "0" in the PUBLISH request. Such a request SHOULD NOT contain any body. UAs which support PUBLISH SHOULD support this mechanism for explicitly removing event state.

8. Querying the Current Event State

The response to a PUBLISH request indicates whether the request was successful or not. In general, the body of a such response will be empty unless the event package defines explicit meaning for such a body. There is no such meaning for a response to a presence publication when the document format used is CPIM PIDF.

To query the event state that the state agent in fact publishes, the client may SUBSCRIBE to the event package for which it has sent a PUBLISH, indicating the same address-of-record in the To header. An Expires header value of "0" may be used in this SUBSCRIBE request to do a one-time fetch of this event state as defined in RFC3265.

9. Refreshing Event State

Each EPA is responsible for refreshing the publications that it has previously established. An EPA MAY choose to refresh the publication established by another EPA for the same address-of-record. The authorization policy of the ESC.

The 200 (OK) response from the state agent MUST contain an Expires header indicating the expiration time interval for the publication. The EPA then issues a PUBLISH request for each of its publications before the expiration interval has elapsed.

If an EPA receives a 423 (Interval Too Brief) response to a PUBLISH request, it MAY retry the publication after changing the expiration interval in the Expires header to be equal to or greater than the expiration interval within the Min-Expires header field of the 423(Interval Too Brief) response.

10. Processing PUBLISH Requests

The Event State Compositor (ESC) is a UAS that responds to PUBLISH requests and maintains a list of publications for a given address-of-record. The ESC MUST ignore the Record-Route header field if it is included in a PUBLISH request. The ESC MUST NOT include a Record-Route header field in any response to a PUBLISH request.

The ESC has to know (for example, through configuration) the set of domain(s) for which it maintains event state. PUBLISH requests MUST be processed in the order that they are received. PUBLISH requests MUST also be processed atomically, meaning that a particular PUBLISH request is either processed completely or not at all.

When receiving a PUBLISH request, the ESC follows these steps:

1. The ESC inspects the Request-URI to determine whether this request is for a domain supported by the ESC. If not, the ESC SHOULD proxy the request to the addressed domain.

2. To guarantee that the ESC supports any necessary extensions, the ESC MUST process the Require header field values as described for UASs in Section 8.2.2 of RFC3261.

3. An ESC SHOULD authenticate the UAC. Mechanisms for the authentication of SIP user agents are described in Section 22 of RFC3261.

4. The ESC SHOULD determine if the authenticated user is authorized to publish for this address-of-record. If the authenticated user is not authorized to publish, the ESC MUST return a 403 (Forbidden). This authorization may take into account 3rd party publication of event state.

5. The ESC extracts the address-of-record from the To header field of the request. If the address-of-record is not valid for the domain in the Request-URI, the ESC MUST send a 404 (Not Found) response and skip the remaining steps. The URI MUST then be converted to a canonical form. To do that, all URI parameters MUST be removed (including the user-param), and any escaped characters MUST be converted to their unescaped form. The result serves as an index into the list of publications.

6. The ESC examines the Event header of the PUBLISH request. If the Event header is missing or contains an event package which the ESC does not support, the ESC MUST respond to the PUBLISH request with a 489 (Bad Event) response.

7. The ESC now processes the Expires header value from the PUBLISH request.

   * If the request has an Expires header field, that value MUST be taken as the requested expiration.

   * Else, a locally-configured default value MUST be taken as the requested expiration.

   * The ESC MAY choose an expiration less than the requested expiration interval. If and only if the requested expiration interval is greater than zero AND less than a ESC-configured minimum, the ESC MAY reject the publication with a response of 423 (Interval Too Brief). This response MUST contain a Min-Expires header field that states the minimum expiration interval the ESC is willing to honor. It then skips the remaining steps.

8. The ESC may then process the body of the PUBLISH request (the actual event state)

   * For each publication, the ESC will record the target of the publication (To URI), the source of the publication (From URI), and the version of the publication. This version information must be present in the body of the PUBLISH request. In the presence publication application, this information will come from the timestamp element associated with each presence tuple.

   * If the version of the event state present in the PUBLISH request is older than the current version known by the ESC, the ESC MUST return a 494 (Out of Sync) response and MUST NOT update the event state for this AOR. This is to handle out-of-order or stale PUBLISH requests. To recover from this error, the client SHOULD determine the current version of the event state at the server by sending a SUBSCRIBE request to the server and re-issue the PUBLISH request if the event state changes again.

   * The processing of the PUBLISH request must be atomic. If internal errors (such as the inability to access a back-end database) occur before processing is complete, no portion of the PUBLISH document must be published and the ESC MUST fail with a 500 (Server Error) response.

9. The ESC returns a 200 (OK) response. The response MUST contain an Expires header indicating the expiration interval chosen by the ESC. The state agent associated with this ESC may then issue

appropriate NOTIFY requests to any watchers of this event state. The timing between the receipt of the PUBLISH request and the issuance of NOTIFY requests is implementation dependent and may vary according to throttling policies at the state agent.

11. Syntax

11.1 New Method

The following is the BNF definition for the PUBLISH method. As with all other SIP methods, the method name is case sensitive.

```
PUBLISHm = %x50.55.42.4C.49.53.48 ; PUBLISH in caps.
```

Tables 1 and 2 extend Tables 2 and 3 of RFC 3261 [1] by adding an additional column, defining the header fields that can be used in PUBLISH requests and responses.

Header Field	where	proxy	PUBLISH
Accept	R		-
Accept	2xx		-
Accept	415		m*
Accept-Encoding	R		-
Accept-Encoding	2xx		-
Accept-Encoding	415		m*
Accept-Language	R		-
Accept-Language	2xx		-
Accept-Language	415		m*
Alert-Info	R		-
Alert-Info	180		-
Allow	R		o
Allow	2xx		o
Allow	r		o
Allow	405		m
Authentication-Info	2xx		o
Authorization	R		o
Call-ID	c	r	m
Call-Info		ar	o
Contact	R		-
Contact	1xx		-
Contact	2xx		-
Contact	3xx		o
Contact	485		o
Content-Disposition			o
Content-Encoding			o
Content-Language			o
Content-Length		ar	t
Content-Type			*

CSeq	c	r	m
Date		a	o
Event	a	m	
Error-Info	300-699	a	o
Expires			o
From	c	r	m
In-Reply-To	R		o
Max-Forwards	R	amr	m
Organization		ar	o

Table 1: Summary of header fields, A--O

Header Field	where	proxy	PUBLISH
Priority	R	ar	o
Proxy-Authenticate	407	ar	m
Proxy-Authenticate	401	ar	o
Proxy-Authorization	R	dr	o
Proxy-Require	R	ar	o
Record-Route		ar	-
Reply-To			o
Require		ar	c
Retry-After	404,413,480,486		o
	500,503		o
	600,603		o
Route	R	adr	o
Server	r		o
Subject	R		o
Timestamp			o
To	c(1)	r	m
Unsupported	420		o
User-Agent			o
Via	R	amr	m
Via	rc	dr	m
Warning	r		o
WWW-Authenticate	401	ar	m
WWW-Authenticate	407	ar	o

Table 2: Summary of header fields, P--Z

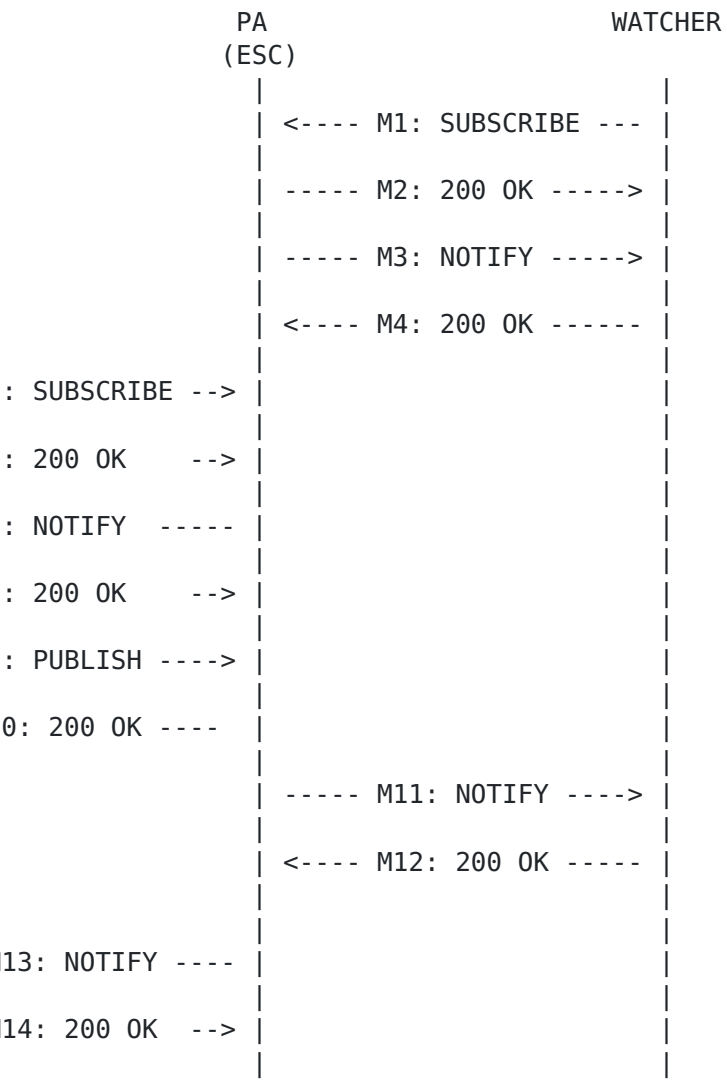
11.2 New Response Code

11.2.1 "494 Out Of Sync" Response Code

The 494 event response is added to the "Client-Error" header field definition. "494 Out of Sync" is used to indicate that the server detected that the event state that the client is trying to publish is out of sync (stale) relative to the event state that the server has. The version information in the PUBLISH body is older than the version information that the server maintains for the corresponding event and AOR.

12. Examples

The following section shows an example of the usage of the PUBLISH method in the case of publishing the presence document from a presence user agent to a presence agent. The watcher in this case is watching the PUA's presentity. The PUA will SUBSCRIBE to its own presence to see the composite presence state exposed by the PA. This is an optional but likely step for the PUA.



Message flow:

M1: The watcher initiates a new subscription to the presentity@domain.com's presence agent.

```
SUBSCRIBE sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP 10.0.0.1:5060;branch=z9hG4bKnashds7
To: <sip:presentity@domain.com>
From: <sip:watcher@domain.com>;tag=12341234
Call-ID: 12345678@10.0.0.1
CSeq: 1 SUBSCRIBE
Expires: 3600
Event: presence
Contact: <sip:watcher@domain.com>
Content-Length: 0
```

M2: The presence agent for presentity@domain.com processes the subscription request and creates a new subscription. A 200 (OK) response is sent to confirm the subscription.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.0.1:5060;branch=z9hG4bKnashds7
To: <sip:presentity@domain.com>;tag=abcd1234
From: <sip:watcher@domain.com>;tag=12341234
Call-ID: 12345678@10.0.0.1
CSeq: 1 SUBSCRIBE
Contact: <sip:pa@domain.com>
Expires: 3600
Content-Length: 0
```

M3: In order to complete the process, the presence agent sends the watcher a NOTIFY with the current presence state of the presentity.

```
NOTIFY sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP pa.domain.com;branch=z9hG4bK8sdf2
To: <sip:watcher@domain.com>;tag=12341234
From: <sip:presentity@domain.com>;tag=abcd1234
Call-ID: 12345678@10.0.0.1
CSeq: 1 NOTIFY
Event: presence
Subscription-State: active; expires=3599
Content-Type: application/cpim-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
          entity="pres:presentity@domain.com">
   <tuple id="mobile-phone">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T16:49:29Z</timestamp>
   </tuple>
   <tuple id="desktop">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T12:21:29Z</timestamp>
   </tuple>
</presence>
```

M4: The watcher confirms receipt of the NOTIFY request.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pa.domain.com;branch=z9hG4bK8sdf2
To: <sip:watcher@domain.com>;tag=12341234
From: <sip:presentity@domain.com>;tag=abcd1234
Call-ID: 12345678@10.0.0.1
CSeq: 1 NOTIFY
Contact: <sip:watcher@domain.com>
```

M5: To view its composite presence state, the PUA issues a SUBSCRIBE to the PA for itself.

```
SUBSCRIBE sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP 10.0.0.2:5060;branch=z9hG4bKjjsdfj
To: <sip:presentity@domain.com>
From: <sip:presentity@domain.com>;tag=43214321
Call-ID: 87654321@10.0.0.2
CSeq: 1 SUBSCRIBE
Expires: 3600
Event: presence
Contact: <sip:pua@domain.com>
Content-Length: 0
```

M6: The presence agent for presentity@domain.com processes the subscription request and creates a new subscription. A 200 (OK) response is sent to confirm the subscription.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.0.2:5060;branch=z9hG4bKjjsdfj
To: <sip:presentity@domain.com>;tag=abcd1235
From: <sip:watcher@domain.com>;tag=43214321
Call-ID: 87654321@10.0.0.2
CSeq: 1 SUBSCRIBE
Contact: <sip:pa@domain.com>
Expires: 3600
Content-Length: 0
```

M7: In order to complete the process, the presence agent sends the PUA a NOTIFY with the current presence state of the presentity.

```
NOTIFY sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP pa.domain.com;branch=z9hG4bK8sdfk
To: <sip:watcher@domain.com>;tag=abcd1235
From: <sip:presentity@domain.com>;tag=43214321
Call-ID: 87654321@10.0.0.2
CSeq: 1 NOTIFY
Event: presence
Subscription-State: active; expires=3599
Content-Type: application/cpim-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
          entity="pres:presentity@domain.com">
   <tuple id="mobile-phone">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T16:49:29Z</timestamp>
   </tuple>
   <tuple id="desktop">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T12:21:29Z</timestamp>
   </tuple>
</presence>
```

M9: A presence user agent for the presentity detects a change in the user's presence state. It initiates a PUBLISH to the presentity's presence agent in order to update it with the new presence information. The timestamp element is updated to indicate the time of the change. The Expires header indicates the desired duration of this soft-state. The "entity" attribute of the presence element in the PIDF document matches the To AOR.

```
PUBLISH sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP pua.domain.com;branch=z9hG4bK652hsge
To: <sip:presentity@domain.com>;tag=1a2b3c4d
From: <sip:presentity@domain.com>;tag=1234wxyz
Call-ID: 81818181@pua.domain.com
CSeq: 1 PUBLISH
Expires: 3600
Event: presence
Content-Type: application/cpim-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
          entity="pres:presentity@domain.com">
   <tuple id="mobile-phone">
      <status>
         <basic>closed</basic>
      </status>
      <timestamp>2003-02-01T17:00:19Z</timestamp>
   </tuple>
</presence>
```

M10: The presence agent receives, and accepts the presence information. The published data is incorporated into the presentity's presence document. A 200 (OK) response is sent to confirm the publication.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pua.domain.com;branch=z9hG4bK652hsge
To: <sip:presentity@domain.com>;tag=1a2b3c4d
From: <sip:presentity@domain.com>;tag=1234wxyz
Call-ID: 81818181@pua.domain.com
CSeq: 1 PUBLISH
Expires: 1800
```

M11: The presence agent determines that a reportable change has been made to the presentity's presence document, and sends another notification to those watching the presentity to update their information regarding the presentity's current presence status.

```
NOTIFY sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP presence.domain.com;branch=z9hG4bK4cd42a
To: <sip:watcher@domain.com>;tag=12341234
From: <sip:presentity@domain.com>;tag=abcd1234
Call-ID: 12345678@10.0.0.1
CSeq: 2 NOTIFY
Event: presence
Subscription-State: active; expires=3400
Content-Type: application/cpim-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
          entity="pres:presentity@domain.com">
   <tuple id="mobile-phone">
      <status>
         <basic>closed</basic>
      </status>
      <timestamp>2003-02-01T17:00:19Z</timestamp>
   </tuple>
   <tuple id="desktop">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T12:21:29Z</timestamp>
   </tuple>
</presence>
```

M12: The watcher confirms receipt of the NOTIFY request.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP presence.domain.com;branch=z9hG4bK4cd42a
To: <sip:watcher@domain.com>;tag=12341234
From: <sip:presentity@domain.com>;tag=abcd1234
Call-ID: 12345678@10.0.0.1
CSeq: 2 NOTIFY
Content-Length: 0
```

M13: The presence agent also sends a NOTIFY to the PUA.

```
NOTIFY sip:presentity@domain.com SIP/2.0
Via: SIP/2.0/UDP presence.domain.com;branch=z9hG4bK4cd42b
To: <sip:watcher@domain.com>;tag=abcd1235
From: <sip:presentity@domain.com>;tag=43214321
Call-ID: 87654321@10.0.0.2
CSeq: 2 NOTIFY
Event: presence
Subscription-State: active; expires=3400
Content-Type: application/cpim-pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
          entity="pres:presentity@domain.com">
   <tuple id="mobile-phone">
      <status>
         <basic>closed</basic>
      </status>
      <timestamp>2003-02-01T17:00:19Z</timestamp>
   </tuple>
   <tuple id="desktop">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T12:21:29Z</timestamp>
   </tuple>
</presence>
```

M14: The PUA confirms receipt of the NOTIFY request.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP presence.domain.com;branch=z9hG4bK4cd42b
To: <sip:watcher@domain.com>;tag=abcd1235
From: <sip:presentity@domain.com>;tag=43214321
Call-ID: 87654321@10.0.0.2
CSeq: 2 NOTIFY
```

13. IANA Considerations

This document registers a new response code. This response code is defined by the following information, which is to be added to the method and response-code sub-registry under http://www.iana.org/assignments/sip-parameters. Response Code Number: 494 Default Reason Phrase: Out Of Sync

14. Security Considerations

The state agent SHOULD authenticate the Event Publication Agent (EPA), and SHOULD apply authorization policies. The composition model makes no assumptions that all input sources for a compositor (ESC) are on the same network, or in the same administrative domain.

The ESC 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 ESC can help limit refreshes of event state. Additional throttling and debounce logic at the ESC is advisable to further reduce the notification traffic produced as a result of a PUBLISH method.

Integrity protection and privacy of the PUBLISH requests can be ensured using the S/MIME mechanisms outlined in section 23 of RFC3261. Integrity protection of the To, From, Call-ID, CSeq, Event, Route, and Expires headers should be done at a minimum.

If the ESC receives a PUBLISH request which is integrity protected using a security association that is not with the ESC (for example, end-to-end S/MIME integrity protection), the state agent coupled with the ESC MUST NOT modify the event state before exposing it to the watchers of this event state in a NOTIFY request(s). This is to preserve the end-to-end integrity of the event state.

15. Open Issues

- Should the version information of the publication request be carried explicitly in a header of the request, or is sufficient to rely on the body for this information?
- Should the segments of event state (presence tuples) be sent in separate PUBLISH requests or is it enough to treat these as implicitly separate publication requests?
- Should the PUBLISH mechanism be overloaded to publish authorization information (ACLs) for the event state as well?
- Does end-to-end S/MIME integrity protection make sense when an event compositor is used?

Normative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. 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", BCP 14, RFC 2119, March 1997.

[4] Campbell, Olson, Peterson, Rosenberg and Stucker, "SIMPLE Presence Publication Requirements", draft-ietf-simple-publish-reqs-00 (work in progress), February 2003.

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

[6] Olson, "A Mechanism for Content Indirection in SIP Messages", draft-olson-sip-content-indirect-mech-01 (work in progress), August 2002.

[7] Rosenberg, "A Presence Event Package for the Session Initiation Protocol (SIP)", draft-ietf-simple-presence-10 (work in progress), January 2003.

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

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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