SIP WG
Internet-Draft
Expires: April 23, 2004

A. Niemi, Ed.
Nokia

October 24, 2003

Session Initiation Protocol (SIP) Extension for Event State Publication
draft-ietf-sip-publish-01

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 mechanism described in this document can be extended to support publication of any event state, for which there exists an appropriate event package. 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.)

Table of Contents

1. Introduction

The focus of this specification is to provide a framework for the publication of event state from a user agent to an entity that is responsible for composing this event state and distributing it to interested parties through the SIP events [1] framework. This specification fills a gap in the SIP events framework to allow for a client to push its event state to the state agent that acts on its behalf.

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

The mechanism described in this document can be extended to support publication of any event state, for which there exists an appropriate event package as defined in [1]. 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 [5], HTTP [6], etc.)

2. Definitions and Document Conventions

In addition to the definitions of RFC 3265 [1] and RFC 3261 [2], this document introduces some new concepts:

Event State: State information for a resource, associated with an event package and an address-of-record.

Event Publication Agent (EPA): The UAC that issues PUBLISH requests to publish event state. For presence, this corresponds to the PUA.

Event State Compositor (ESC): The UAS that processes PUBLISH requests, and is responsible of composing event state into a complete, composite event state of a resource. For presence, this corresponds to the PA.

Publication: The act of an EPA sending a PUBLISH message to an ESC to publish event state.

Hard State: The steady-state or default event state of a resource, which the ESC may use in the absence of, or in addition to, soft state publications.

Soft State: Event state published by an EPA using the PUBLISH mechanism. A protocol element (i.e., an entity-tag) is used to identify a specific soft state entity in the ESC. Soft state has a defined lifetime and will expire after a negotiated amount of time.

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] and indicate requirement levels for compliant implementations.

Indented passages such as this one are used in this document to provide additional information and clarifying text. They do not contain normative protocol behavior.

3. Overall Operation

This document defines a new SIP method, PUBLISH, for publishing event state. PUBLISH is analogous to REGISTER in that it allows a user to create, modify, and remove state in another entity which manages this state on behalf of the user. The user may in turn have multiple UAs or endpoints. Each endpoint may publish its own unique state, out of which the event agent generates the composite event state of the resource. Through a subscription to that event package, the user is able to discover the composite event state of all the active endpoints.

In the generic sense, a UAC that publishes event state is labeled an Event Publication Agent (EPA). For presence in particular, this is the familiar PUA role as defined in [7]. The entity that processes the PUBLISH request is known as an Event State Compositor (ESC). For presence in particular, this is the familiar PA role as defined in [7].

PUBLISH requests create soft state in the ESC. This state has a defined lifetime and will expire after a negotiated amount of time, requiring the publication to be refreshed by subsequent PUBLISH requests. Local policy at the compositor may in turn define hard state for a particular event package. That is, the steady-state of this event package in the absence of, or in addition to, soft state provided through the PUBLISH mechanism. Setting this hard state or configuring the composer policy is out of the scope of this specification.

Typically, the body of a PUBLISH request carries the published event state. In the response to a PUBLISH request, the ESC assigns an identifier to this event state. This identifier is then used by the

EPA as a request precondition to point to that specific event state in the subsequent PUBLISH requests that modify, refresh or remove that event state. In case the PUBLISH request points to an expired event state, this precondition will fail.

4. Considerations for Event Packages using PUBLISH

This section discusses several issues which should be taken into consideration when applying the PUBLISH mechanism to event packages. It also demonstrates how these issues are handled when using PUBLISH for presence publication.

4.1 PUBLISH Bodies

The body of the PUBLISH request typically carries the published event state. Any application of the PUBLISH mechanism for a given event package MUST define what content type or types are expected in PUBLISH requests. Each event package MUST also describe the semantics associated with that content, and MUST prescribe a default, mandatory to implement MIME type.

This document defines the semantics of the presence publication requests (event package "presence") when the CPIM PIDF [8] presence document format is used. A PUA that uses PUBLISH to publish presence state to the PA MUST support the CPIM PIDF presence format. It MAY support other formats.

4.2 PUBLISH Response Bodies

The response to a PUBLISH request indicates whether the request was successful or not. In general, the body of such a response will be empty unless the event package defines explicit meaning for such a body.

There is no such meaning for the body of a response to a presence publication when the document format used is CPIM PIDF.

4.3 Partial Event State

The content type MUST provide a way to publish partial state for an event package. The intention is to allow each endpoint for an address-of-record to publish event state independently. To accomplish this, the event state that is published by these endpoints MUST be allowed to be only a portion of the complete state that the state agent advertises for that address-of-record.

Note that sources for event state other than those using the PUBLISH mechanism are explicitly allowed. It is beyond the scope

of this document to define such interfaces.

For presence in particular, a PUA can publish presence state for just a subset of the tuples that may be composed 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 and out of the scope of this specification.

4.4 Default Expiration of PUBLISH

PUBLISH requests establish soft state in the ESC which expires after a negotiated amount of time. Each event package MUST provide a default expiration value recommendation (SHOULD strength).

For presence publication, it is RECOMMENDED that the ESC use a default value of 3600 seconds (1 hour) for this default expiration value.

5. Constructing PUBLISH Requests

PUBLISH requests create, modify, and remove event state associated with an address-of-record. A suitably authorized third party may also perform publication on behalf of a particular address-of-record.

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 [2].

If necessary, clients may probe for the support of PUBLISH using the OPTIONS request defined in SIP [2]. The presence of "PUBLISH" in the "Allow" header field in a response to an OPTIONS request indicates support for the PUBLISH method. In addition, the "Allow-Events" header field indicates the supported event packages.

> Note that it is possible for the OPTIONS request to fork, and consequently return a response from a UA other than the ESC. In that case, support for the PUBLISH method may not be appropriately represented for that particular Request-URI.

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 RFC 3261 [2]. 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.

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

5.1 Identification of Published Event State

Identification of published events is provided by four pieces of information: Request-URI, event type, and (optionally) an entity-tag and the message body.

The Request-URI of a PUBLISH request contains enough information to route the request to the appropriate entity per the request routing procedures outlined in SIP [2]. It also contains enough information to identify the resource whose event state is to be published, but not enough information to determine the type of the published event state.

For determining the type of the published event state, the EPA MUST include a single Event header field in the PUBLISH requests. The value of this header field indicates the event package, for which this request is publishing event state.

To update previously published event state, PUBLISH requests MAY contain a single If-Match header field identifying the specific entity of event state that the request is refreshing, modifying or removing. This header field MUST contain a single entity-tag provided by the ESC in receipt of the initial publication.

The PUBLISH request MAY contain a body, which contains event state that the client wishes to publish. The content format and semantics are dependent on the event package identified in the Event header field.

As with any other SIP message, the PUBLISH mechanism MAY use the content indirection mechanism defined in [9]. 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 reasonably be carried directly within the SIP signaling (PUBLISH request).

5.2 Creating Initial Publication

The PUBLISH request created by the EPA and sent to the ESC establishes soft state for the event package indicated in the Event header field of the request, and bound to the address-of-record in the To header field of the request.

The published event state is typically carried in the body of the

PUBLISH request.

The EPA MAY send subsequent PUBLISH requests to refresh, modify, or remove the event state established by a prior publication. These operations will be described in subsequent sections.

An initial PUBLISH request MUST not contain an If-Match header field. However, if the EPA expects an appropriate, locally stored entity-tag to still be valid, it SHOULD try to modify that event state as described in Section 5.5, instead of submitting an initial publication.

5.3 Setting the Expiration Interval

PUBLISH requests SHOULD contain a single Expires header field. This value indicates the suggested lifetime of the event state being published by this request. The actual duration of the soft state is defined by local policy at the ESC.

> For example, a reasonable implementation might maintain event state over a short grace period even after the publication on which it arrived has expired.

If an Expires header is not present, the EPA is indicating its desire for the ESC to choose. The Expires header field in a 200 (OK) response to PUBLISH indicates the actual duration for which the PUBLISH will remain active, unless it is refreshed.

5.4 Refreshing Event State

Each EPA is responsible for refreshing the publications that it has previously established.

The 200 (OK) response to a PUBLISH request from the ESC contains an Expires header field indicating the expiration time interval for the publication. To refresh its publications, the EPA issues a PUBLISH request for each of its publications before the expiration interval has elapsed.

Also, the 200 (OK) response to a PUBLISH request from the ESC contains an ETag header field with a single entity-tag indicating an identifier for the published event state. To refresh the event state, the EPA includes the received entity-tag in an If-Match header field of the PUBLISH request.

> Note that for the EPA, the entity-tag is simply an opaque token without any semantics associated with it.

The If-Match header field containing an entity-tag preconditions the PUBLISH request to refresh a specific instance of event state in the ESC. If the entity-tag matches a valid event state in the ESC, the refresh is successful, and the EPA receives a 200 (OK) response. If there is no matching event state at the ESC, i.e., the event state to be refreshed has already expired, the EPA receives a 412 (Precondition Failed) response to the PUBLISH request.

A PUBLISH request that refreshes event state SHOULD NOT contain a body.

5.5 Modifying Event State

Modifying event state closely resembles the creation of initial event state. But instead of establishing completely new event state in the ESC, already existing event state is replaced with modified event state. Typically, the modified event state is carried in the body of the PUBLISH request.

The 200 (OK) response to a PUBLISH request from the ESC contains an ETag header field with a single entity-tag indicating an identifier for the published event state. To modify that event state, the EPA includes the received entity-tag in an If-Match header field of the PUBLISH request.

The If-Match header field containing an entity-tag preconditions the PUBLISH request to modify a specific instance of event state in the ESC. If the entity-tag matches a valid event state in the ESC, that event state is replaced by the event state carried in the PUBLISH request, and the EPA receives a 200 (OK) response. If there is no matching event state at the ESC, i.e., the event state to be modified has already expired, the EPA receives a 412 (Precondition Failed) response to the PUBLISH request.

Note that the entity-tag will remain unchanged when modifying the event state associated with it.

5.6 Removing Event State

Soft state established by the PUBLISH request will expire unless periodically refreshed. This event state may also be explicitly removed. An EPA can influence the expiration interval selected by the ESC as described in Section 5.3.

An EPA requests the immediate removal of event state by specifying in the PUBLISH request an Expires value of "0", and setting the If-Match header field to contain the entity-tag of the event state to be

removed.

Note that removing event state is effectively a publication refresh suggesting an infinitesimal expiration interval. Consequently, the refreshed event state expires immediately after being refreshed.

A PUBLISH request that removes event state SHOULD NOT contain a body. EPAs which support the PUBLISH method SHOULD support this mechanism for explicitly removing event state.

5.7 Querying the Current Event State

To query the composite event state that the state agent in fact delivers to the subscribers, 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 [1].

Note that a subscription to the event package will likely deliver results of the event composition process of the state agent, which may be a subset or a superset of the current published event state.

5.8 Error Responses

If an EPA receives a 412 (Precondition Failed) response, it MUST NOT reattempt the PUBLISH request. Instead, to publish event state, the EPA SHOULD perform an initial publication, i.e., a PUBLISH request without a request precondition, as described in Section 5.2.

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 field to be equal to or greater than the expiration interval within the Min-Expires header field of the 423 (Interval Too Brief) response.

6. Processing PUBLISH Requests

The Event State Compositor (ESC) is a UAS that processes and responds to PUBLISH requests, and maintains a list of publications for a given address-of-record. The ESC has to know (e.g., through configuration) the set of addresses for which it maintains event state.

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.

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.

A client may probe the ESC for the support of PUBLISH using the OPTIONS request defined in SIP [2]. In the response to such an OPTIONS request, the ESC SHOULD include "PUBLISH" to the list of allowed methods in the Allow header field. Also, it SHOULD list the supported event packages in an Allow-Events header field.

> The "methods" Contact header field parameter may also be used to specifically announce support for PUBLISH messages when registering. (See SIP Capabilities [10] for details on the "methods" parameter).

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

1. The ESC inspects the Request-URI to determine whether this request is targeted to a resource for which the ESC is responsible for maintaining event state. If not, the ESC MUST return a 404 (Not Found) response and skip the remaining steps.

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

3. An ESC SHOULD authenticate the EPA. Mechanisms for the authentication of SIP user agents are described in Section 22 of RFC3261 [2]. If no authentication mechanism is available, the ESC MAY take the address-of-record of the From header field as the asserted identity of the originator of the request.

4. The ESC SHOULD determine if the authenticated user is authorized to perform event state publication for the identified by the Request-URI. If the authenticated user is not authorized, the ESC MUST return a 403 (Forbidden) response and skip the rest of the remaining steps.

   Note that this authorization may need to take into account third-party publication of event state.

5. The ESC examines the Event header field of the PUBLISH request. If the Event header field 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, and skip the

remaining steps.

6. The ESC examines the If-Match header field of the PUBLISH request for the presence of a request precondition.

   * If the request has an If-Match header field, the ESC checks whether the header field contains a single entity-tag. If not, the request is invalid, and the ESC MUST return with a 400 (Invalid Request) response and skip the remaining steps.

   * Else, the ESC extracts the entity-tag contained in the If-Match header field and matches that entity-tag against all locally stored entity-tags for this resource and event package. If no match is found, the ESC MUST reject the publication with a response of 412 (Precondition Failed), and skip the remaining steps.

   * If the request contains no If-Match header field, the ESC MUST generate and store a locally unique entity-tag for identifying the publication.

     Note that the exact way in which the ESC creates the entity-tag is a matter of local policy. One reasonable implementation of an entity-tag is a counter which is incremented by one each time a publication is allocated a new entity-tag.

7. The ESC processes the Expires header field 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. Only if the requested expiration interval is greater than zero and less than a locally-configured minimum, the ESC MAY reject the publication with a response of 423 (Interval Too Brief), and skip the remaining steps. This response MUST contain a Min-Expires header field that states the minimum expiration interval the ESC is willing to honor.

8. The ESC processes the published event state, typically contained in the body of the PUBLISH request. If the request contains no body (when it should contain one), or the content type of the request does not match the event package, or is not understood by

the ESC, the ESC MUST reject the request with an appropriate response and skip the remainder of the steps.

* For each publication, the ESC records the target of the publication, the entity-tag identifying the publication, the expiration value of the event state, and a pointer to the actual event state.

* If present, the ESC stores the event state delivered in the PUBLISH request and identified by the associated entity-tag.

* Else, the event state identified by the entity-tag is refreshed, setting the expiration value to the chosen expiration interval. If the chosen expiration interval has a special value of "0", the event state identified by the entity-tag MUST be immediately removed.

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, the publication MUST NOT succeed, 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 response MUST also contain an ETag header identifying the published event state. The state agent associated with this ESC may then issue appropriate NOTIFY requests to any watchers of this event state.

   Note that the timing between the receipt of the PUBLISH request and the issuance of NOTIFY requests is implementation dependent and may also vary according to throttling policies at the state agent.

7. Syntax

This section describes the syntax extensions required for event publication in SIP. The formal syntax definitions described in this section are expressed in the Augmented BNF format used in SIP [2], and contain references to elements defined therein.

7.1 New Methods

7.1.1 PUBLISH Method

"PUBLISH" is added to the definition of the element "Method" in the

SIP message grammar. As with all other SIP methods, the method name is case sensitive. PUBLISH is used to publish event state to an entity responsible for composing this event state.

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

Header Field	where	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		-
Allow	R	o
Allow	2xx	o
Allow	r	o
Allow	405	m
Allow-Events	R	o
Allow-Events	489	m
Authentication-Info	2xx	o
Authorization	R	o
Call-ID	c	m
Call-Info		o
Contact	R	-
Contact	1xx	-
Contact	2xx	-
Contact	3xx	o
Contact	485	o
Content-Disposition		o
Content-Encoding		o
Content-Language		o
Content-Length		t
Content-Type		*
CSeq	c	m
Date		o
Event	R	m
Error-Info	300-699	o
Expires		o
Expires	2xx	m
From	c	m
In-Reply-To	R	-
Max-Forwards	R	m
Min-Expires	423	m
MIME-Version		o
Organization		o

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

Header Field	where	PUBLISH
Priority	R	o
Proxy-Authenticate	407	m
Proxy-Authenticate	401	o
Proxy-Authorization	R	o
Proxy-Require	R	o
Record-Route		-
Reply-To		-
Require		o
Retry-After	404,413,480,486	o
Retry-After	500,503	o
Retry-After	600,603	o
Route	R	c
Server	r	o
Subject	R	o
Supported	R	o
Supported	2xx	o
Timestamp		o
To	c(1)	m
Unsupported	420	o
User-Agent		o
Via	R	m
Via	rc	m
Warning	r	o
WWW-Authenticate	401	m
WWW-Authenticate	407	o

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

7.2 New Response Codes

7.2.1 "412 Precondition Failed" Response Code

The 412 (Precondition Failed) response is added to the "Client-Error" header field definition. 412 (Precondition Failed) is used to indicate that the precondition given for the request has failed.

7.3 New Header Fields

Table 3 expands on Table 2 in SIP [2], as amended by the changes in Section 7.1.

Header Field	whr	prx	AC	BY	CA	IN	OP	RE	PUB
ETag	2xx		-	-	-	-	-	-	m
If-Match	R		-	-	-	-	-	-	o

Table 3: Summary of header fields, A--O

7.3.1 "ETag" Header Field

ETag is added to the definition of the element "general-header" in the SIP message grammar. Usage of this header is described in Section 6.

7.3.2 "If-Match" Header Field

If-Match is added to the definition of the element "general-header" in the SIP message grammar. Usage of this header is described in Section 5.

7.4 Augmented BNF Definitions

This section describes the Augmented BNF definitions for the various new and modified syntax elements. The notation is as used in SIP [2] and the documents to which it refers.

```
PUBLISHm            = %x50.55.42.4C.49.53.48 ; PUBLISH in caps.
extension-method    = PUBLISHm / token
ETag                = "ETag" HCOLON entity-tag
If-Match            = "If-Match" HCOLON entity-tag
entity-tag          = token
```

8. IANA Considerations

This document registers a new method name, a new response code and two new header field names.

8.1 Methods

This document registers a new SIP method, 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.

Method Name: PUBLISH

       Reference: [RFCYYYY]

      (Note to RFC Editor: Replace YYYY with the RFC number of this
      document when published).

8.2 Response Codes

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:	412
Default Reason Phrase:	Precondition Failed

8.3 Header Field Names

This document registers two new SIP header field names. These headers are defined by the following information, which is to be added to the header sub-registry under http://www.iana.org/assignments/sip-parameters.

Header Name:	ETag
Compact Form:	(none)

Header Name:	If-Match
Compact Form:	(none)

9. Security Considerations

9.1 Access Control

Since event state may be considered sensitive information, the ESC should have the ability to selectively accept publications from authorized sources only, based on the identity of the EPA.

The state agent SHOULD authenticate the EPA, and SHOULD apply its authorization policies (e.g., based on access control lists) to all requests. The composition model makes no assumptions that all input sources for an ESC are on the same network, or in the same administrative domain.

Authentication issues are discussed in SIP [2]. The exact methods for creation and manipulation of the ESC authorization policies are outside the scope of this document.

9.2 Denial of Service Attacks

The creation of state at the ESC upon receipt of a PUBLISH request can be used by attackers to consume resources on a victim's machine, possibly rendering it unusable.

To reduce the chances of such an attack, implementations of ESCs SHOULD require authentication of PUBLISH requests. Authentication issues are discussed in SIP [2].

Also, 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 header field values for the supported event packages at an 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 request.

9.3 Replay Attack

Replaying the PUBLISH request can have detrimental effects. An attacker may be able to perform any event state publication it witnessed being performed at some point in the past, by replaying a PUBLISH request. Among other things, such a replay message may be used to spoof old event state information, although a versioning mechanism, e.g., a timestamp, in the state information may help mitigate such an attack.

To prevent replay attacks, implementations SHOULD require authentication with anti-replay protection. Authentication issues are discussed in SIP [2].

9.4 Man in the Middle Attacks

Even with authentication, man-in-the-middle attacks using PUBLISH may be used to install arbitrary event state information, modify or remove existing event state information in publications, or even remove event state altogether at an ESC.

To prevent such attacks, implementations SHOULD, at a minimum, provide integrity protection across the To, From, Event, If-Match, Route, and Expires headers and the bodies of PUBLISH messages.

If the ESC receives event state in a PUBLISH request which is integrity protected using a security association that is not with the ESC (e.g., integrity protection is applied end-to-end, from publisher to subscriber), the state agent coupled with the ESC MUST NOT modify the event state before exposing it to the subscribers of this event

state in NOTIFY requests. This is to preserve the end-to-end integrity of the event state.

Integrity protection of message headers and bodies is discussed in SIP [2].

9.5 Confidentiality

The state information contained in a PUBLISH message may potentially contain sensitive information. Implementations MAY encrypt such information to ensure confidentiality.

The mechanisms for providing confidentiality are detailed in SIP [2].

10. Examples

This 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 may also 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, and is not shown in this example.

TBD: replace domain.com with example.com

```
          PUA                     PA                      WATCHER
         (EPA)                   (ESC)
           |                       |                         |
           |                       | <---- M1: SUBSCRIBE --- |
           |                       |                         |
           |                       | ----- M2: 200 OK -----> |
           |                       |                         |
           |                       | ----- M3: NOTIFY -----> |
           |                       |                         |
           |                       | <---- M4: 200 OK ------ |
           |                       |                         |
           |                       |                         |
           | ---- M5: PUBLISH ---> |                         |
           |                       |                         |
           | <--- M6: 200 OK ----  |                         |
           |                       |                         |
           |                       | ----- M7: NOTIFY -----> |
           |                       |                         |
           |                       | <---- M8: 200 OK ------ |
           |                       |                         |
           | ---- M9: PUBLISH ---> |                         |
           |                       |                         |
           | <--- M10: 200 OK ---  |                         |
           |                       |                         |
           |                       |                         |
           | --- M11: PUBLISH ---> |                         |
           |                       |                         |
           | <-- M12: 200 OK ----  |                         |
           |                       |                         |
           |                       | ----- M13: NOTIFY ----> |
           |                       |                         |
           |                       | <---- M14: 200 OK ----- |
           |                       |                         |
```

Message flow:

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

```
      SUBSCRIBE sip:presentity@example.com SIP/2.0
      Via: SIP/2.0/UDP 10.0.0.1:5060;branch=z9hG4bKnashds7
      To: <sip:presentity@example.com>
      From: <sip:watcher@example.com>;tag=12341234
      Call-ID: 12345678@10.0.0.1
      CSeq: 1 SUBSCRIBE
      Max-Forwards: 70
      Expires: 3600
      Event: presence
      Contact: <sip:watcher@example.com>
      Content-Length: 0
```

M2: The presence agent for presentity@example.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@example.com>;tag=abcd1234
      From: <sip:watcher@example.com>;tag=12341234
      Call-ID: 12345678@10.0.0.1
      CSeq: 1 SUBSCRIBE
      Contact: <sip:pa@example.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@example.com SIP/2.0
      Via: SIP/2.0/UDP pa.example.com;branch=z9hG4bK8sdf2
      To: <sip:watcher@example.com>;tag=12341234
      From: <sip:presentity@example.com>;tag=abcd1234
      Call-ID: 12345678@10.0.0.1
      CSeq: 1 NOTIFY
      Max-Forwards: 70
      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@example.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.example.com;branch=z9hG4bK8sdf2
      To: <sip:watcher@example.com>;tag=12341234
      From: <sip:presentity@example.com>;tag=abcd1234
      Call-ID: 12345678@10.0.0.1
      CSeq: 1 NOTIFY
      Contact: <sip:watcher@example.com>
```

M5: A presence user agent for the presentity initiates a PUBLISH to the presentity's presence agent in order to update it with new presence information. The Expires header indicates the desired duration of this soft state.

```
      PUBLISH sip:presentity@example.com SIP/2.0
      Via: SIP/2.0/UDP pua.example.com;branch=z9hG4bK652hsge
      To: <sip:presentity@example.com>
      From: <sip:presentity@example.com>;tag=1234wxyz
      Call-ID: 81818181@pua.example.com
      CSeq: 1 PUBLISH
      Max-Forwards: 70
      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@example.com">
         <tuple id="mobile-phone">
            <status>
               <basic>closed</basic>
            </status>
            <timestamp>2003-02-01T17:00:19Z</timestamp>
         </tuple>
      </presence>
```

M6: 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. The 200 (OK) response contains an ETag header field with an entity-tag. This is used to identify the published event state in subsequent PUBLISH requests.

```
      SIP/2.0 200 OK
      Via: SIP/2.0/UDP pua.example.com;branch=z9hG4bK652hsge
      To: <sip:presentity@example.com>;tag=1a2b3c4d
      From: <sip:presentity@example.com>;tag=1234wxyz
      Call-ID: 81818181@pua.example.com
      CSeq: 1 PUBLISH
      ETag: dx200xyz
      Expires: 1800
```

M7: 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@example.com SIP/2.0
      Via: SIP/2.0/UDP presence.example.com;branch=z9hG4bK4cd42a
      To: <sip:watcher@example.com>;tag=12341234
      From: <sip:presentity@example.com>;tag=abcd1234
      Call-ID: 12345678@10.0.0.1
      CSeq: 2 NOTIFY
      Max-Forwards: 70
      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@example.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>
```

M8: The watcher confirms receipt of the NOTIFY request.

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

M9: The PUA determines that the event state it previously published is about to expire, and refreshes that event state.

```
      PUBLISH sip:presentity@example.com SIP/2.0
      Via: SIP/2.0/UDP pua.example.com;branch=z9hG4bK771ash02
      To: <sip:presentity@example.com>
      From: <sip:presentity@example.com>;tag=1234kljk
      Call-ID: 98798798@pua.example.com
      CSeq: 1 PUBLISH
      Max-Forwards: 70
      If-Match: dx200xyz
      Expires: 3600
      Event: presence
      Content-Length: 0
```

M10: The presence agent receives, and accepts the publication refresh. The timers regarding the expiration of the specific event state identified by the entity-tag are updated. Note that no actual state change has occured, so the watchers will receive no NOTIFYs.

```
      SIP/2.0 200 OK
      Via: SIP/2.0/UDP pua.example.com;branch=z9hG4bK771ash02
      To: <sip:presentity@example.com>;tag=2affde434
      From: <sip:presentity@example.com>;tag=1234kljk
      Call-ID: 98798798@pua.example.com
      CSeq: 1 PUBLISH
      ETag: dx200xyz
      Expires: 1800
```

M11: The PUA of the presentity detects a change in the user's presence state. It initiates a PUBLISH request to the presence agent to modify the published presence information with the recent change.

```
      PUBLISH sip:presentity@example.com SIP/2.0
      Via: SIP/2.0/UDP pua.example.com;branch=z9hG4bKcdad2
      To: <sip:presentity@example.com>
      From: <sip:presentity@example.com>;tag=54321mm
      Call-ID: 5566778@pua.example.com
      CSeq: 1 PUBLISH
      Max-Forwards: 70
      If-Match: dx200xyz
      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@example.com">
         <tuple id="mobile-phone">
            <status>
               <basic>open</basic>
            </status>
            <timestamp>2003-02-01T19:15:15Z</timestamp>
         </tuple>
      </presence>
```

M12: The presence agent receives, and accepts the publication modification. The timers regarding the expiration of the specific event state identified by the entity-tag are updated, and the published data is incorporated into the presentity's presence document. Note that the document delivered in this modification will replace the previous document.

```
      SIP/2.0 200 OK
      Via: SIP/2.0/UDP pua.example.com;branch=z9hG4bKcdad2
      To: <sip:presentity@example.com>;tag=effe22aa
      From: <sip:presentity@example.com>;tag=54321mm
      Call-ID: 5566778@pua.example.com
      CSeq: 1 PUBLISH
      ETag: dx200xyz
      Expires: 3600
```

M13: 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@example.com SIP/2.0
      Via: SIP/2.0/UDP presence.example.com;branch=z9hG4bK32defd3
      To: <sip:watcher@example.com>;tag=12341234
      From: <sip:presentity@example.com>;tag=abcd1234
      Call-ID: 12345678@10.0.0.1
      CSeq: 2 NOTIFY
      Max-Forwards: 70
      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@example.com">
         <tuple id="mobile-phone">
            <status>
               <basic>open</basic>
            </status>
            <timestamp>2003-02-01T19:15:15Z</timestamp>
         </tuple>
         <tuple id="desktop">
            <status>
               <basic>open</basic>
            </status>
            <timestamp>2003-02-01T12:21:29Z</timestamp>
         </tuple>
      </presence>
```

M14: The watcher confirms receipt of the NOTIFY request.

```
      SIP/2.0 200 OK
      Via: SIP/2.0/UDP presence.example.com;branch=z9hG4bK32defd3
      To: <sip:watcher@example.com>;tag=12341234
      From: <sip:presentity@example.com>;tag=abcd1234
      Call-ID: 12345678@10.0.0.1
      CSeq: 2 NOTIFY
      Content-Length: 0
```

11. Contributors

The original contributors to this specification are:

Ben Campbell
dynamicsoft

Sean Olson

Microsoft

Jon Peterson
Neustar, Inc.

Jonathan Rosenberg
dynamicsoft

Brian Stucker
Nortel Networks, Inc.

12. Document Change History

(Note to RFC Editor: please remove this whole section prior to publication as an RFC.)

12.1 Changes since "draft-ietf-sip-publish-00"

The following changes were made since the last version:

- o Specified the role of the Request-URI in identifying the publication target resource. Also, clarified chapter 5 in this regard to explicitly talk about the identification of publications.
- o Changed chapter 6 to use Request-URI in determining the publication target resource. Also clarified language within the processing steps of an ESC.
- o Added missing header fields and removed unneeded "proxy" column in Table 1 and Table 2. Corrected Table 3 content.
- o Corrected various nits in examples and in body text.

12.2 Changes since "draft-ietf-simple-publish-01"

The following changes were made since the last version:

- o Submitted as "draft-ietf-sip-publish-00".
- o Changed title to better reflect the content.
- o Removed event state segmentation and collision detection of segments, and simplified usage of entity-tags.
- o Rewrote Ch 4 "Considerations for Event Packages Using PUBLISH" to

mimic the way RFC 3265 defines considerations for event packages. Also, removed normative dependency to "draft-ietf-simple-publish-reqs".

- Rewrote Ch 9 "Security Considerations" to now include text about specific vulnerabilities and the security tools to counter those attacks.

- Clarified both UAC and UAS usage of entity-tags. Moved common error handling of UACs to a separate sub-section.

- Improved description of UAS functionality of Ch 6 "Processing PUBLISH Requests", and alinged it with RFC 3261 Chapter 10 on processing registrations.

- Changed entity-tag syntax from "quoted-string" to "token". This is a deviation from RFC 2616 entity-tag syntax, but more aligned to how similar things are expressed in SIP.

- Restricted the If-Match header syntax to only allow a single entity-tag. Multiple entity-tags are not applicable to PUBLISH.

- Added methods other than PUBLISH to Table 3.

- Rewrote Ch 10 "Examples" to better reflect actual PUBLISH usage.

- Changed reference [10] from caller-prefs to callee-caps.

- Overall language and structure tweaking.

12.3 Changes since "draft-ietf-simple-publish-00"

The following changes were made since the last version:

- Merged with "draft-olson-simple-publish-02"

- Removed usage of Call-ID and CSeq for ordering

- Removed timestamp based versioning

- Added versioning based on entity-tag version information (ETag), and request precondition (If-Match)

- Changed reference to content-indirection as Informative

- Added section for ABNF definitions

- o Editorial corrections, restructuring of document to improve readability
- o Moved the original authors into a new "Contributors" section
- o Added new definitions in Terminology, and clarified EPA and ESC definitions
- o Strengthened the IANA considerations section.
- o Added text for announcing/probing support for publish, namely OPTIONS and "methods" parameter usage.

Normative References

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

[2] 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.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Informative References

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

[5] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[6] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

[8] Sugano, H. and S. Fujimoto, "Presence Information Data Format (PIDF)", draft-ietf-impp-cpim-pidf-08 (work in progress), May 2003.

[9] Olson, S., "A Mechanism for Content Indirection in SIP Messages", draft-olson-sip-content-indirect-mech-01 (work in

progress), August 2002.

[10] Rosenberg, J., "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", draft-ietf-sip-callee-caps-00 (work in progress), June 2003.

Author's Address

Aki Niemi (editor)
Nokia
P.O. Box 321
NOKIA GROUP, FIN 00045
Finland

Phone: +358 50 389 1644
EMail: aki.niemi@nokia.com

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

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