SIP WG   A. Niemi, Ed.
Internet-Draft   Nokia
Expires: November 26, 2004   May 28, 2004

           **An Event State Publication Extension to the Session Initiation
                              Protocol (SIP)
                       draft-ietf-sip-publish-04**

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 SIP Events
   framework. The first application of this extension is for 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.

Table of Contents

1. **Introduction**

   This specification provides a framework for the publication of event
   state from a user agent to an entity that is responsible for
   compositing this event state and distributing it to interested
   parties through the SIP Events [1] framework.

   In addition to defining an event publication framework, this
   specification defines a concrete usage of that framework for the
   publication of presence state [2] by a presence user agent [3] to a
   presence compositor, which has a tightly coupled relationship with
   the presence agent [1].

   The requirements and model for presence publication are documented in
   [10]. 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]. For instance, an application of SIP
   events for message waiting indications [11] might choose to collect
   the statuses of voice-mail boxes across a set of user agents using
   the PUBLISH mechanism. The compositor in such an application would
   then be responsible for collecting and distributing this state to the
   subscribers of the event package.

   Each application that makes use of the PUBLISH mechanism in the
   publication of event state will need to adhere to the guidelines set
   in Section 10. The mechanism described in this document is not
   intended to be a general-purpose mechanism for transport of arbitrary
   data, as there are better-suited mechanisms for this purpose.

2. **Definitions and Document Conventions**

   In addition to the definitions of RFC 2778 [3], RFC 3265 [1], and RFC
   3261 [4], 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 User Agent Client (UAC) that
      issues PUBLISH requests to publish event state.

   Event State Compositor (ESC): The User Agent Server (UAS) that
      processes PUBLISH requests, and is responsible for compositing
      event state into a complete, composite event state of a resource.

   Presence Compositor: A type of Event State Compositor that is
      responsible for compositing presence state for a presentity.

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

   Event 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.

   Event 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 at 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 BCP 14, RFC 2119 [5]
   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 descriptions of normative protocol behavior.

3. **Overall Operation**

   This document defines a new SIP method, PUBLISH, for publishing event
   state. PUBLISH is similar 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. Addressing a PUBLISH request is
   identical to addressing a SUBSCRIBE request. The Request-URI of a
   PUBLISH request is populated with the address of the resource for
   which the user wishes to publish event state. The user may in turn
   have multiple User Agents or endpoints that publish event state.
   Each endpoint may publish its own unique state, out of which the
   event state compositor generates the composite event state of the
   resource. In addition to a particular resource, all published event
   state is associated with a specific event package. Through a
   subscription to that event package, the user is able to discover the
   composite event state of all of the active publications.

   A User Agent Client (UAC) that publishes event state is labeled an
   Event Publication Agent (EPA). For presence, this is the familiar
   Presence User Agent (PUA) role as defined in [2]. The entity that
   processes the PUBLISH request is known as an Event State Compositor
   (ESC). For presence, this is the familiar Presence Agent (PA) role

as defined in [2].

PUBLISH requests create soft state in the ESC. This event soft state
has a defined lifetime and will expire after a negotiated amount of
time, requiring the publication to be refreshed by subsequent PUBLISH
requests. There may also be event hard state provisioned for each
resource for a particular event package. This event state represents
the resource state that is present at all times, and does not expire.
The ESC may use event hard state in the absence of, or in addition
to, event soft state provided through the PUBLISH mechanism. Setting
this event hard state or configuring the ESC policy regarding the
aggregation of different event state is out of the scope of this
specification.

The body of a PUBLISH request carries the published event state. In
the response to every successful PUBLISH request, the ESC assigns an
identifier to the publication in the form of an entity-tag. This
identifier is then used by the EPA in any subsequent PUBLISH request
that modifies, refreshes or removes the event state of that
publication. When event state expires or is explicitly removed, the
entity-tag associated with it becomes invalid. A publication for an
invalid entity-tag will naturally fail, and the EPA needs to start
anew and resend its event state without referencing a previous
entity-tag.

4. 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 are identical to the
general UAC behavior described in Section 8.1 and Section 17.1 of RFC
3261 [4].

If necessary, clients may probe for the support of PUBLISH using the
OPTIONS request defined in SIP [4]. 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 User Agent 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 [4]. 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 MAY contain a Contact header field, but including
one in a PUBLISH request has no meaning in the event publication
context and will be ignored by the ESC. An EPA MAY send a PUBLISH
request within an existing dialog. In that case, the request is
received in the context of any media session or sessions associated
with that dialog.

   Note that while sending a PUBLISH request within an existing
   dialog is not prohibited, it will typically not result in the
   expected behavior. Unless the other end of the dialog is also an
   ESC, it will probably reject the request.

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

4.1 Identification of Published Event State

Identification of published event state is provided by three pieces
of information: Request-URI, event type, and (optionally) an
entity-tag.

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 RFC 3261 [4]. 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 PUBLISH requests. The value
of this header field indicates the event package for which this
request is publishing event state.

For each successful PUBLISH request, the ESC will generate and assign
an entity-tag and return it in the SIP-ETag header field of the 2xx
response.

When updating previously published event state, PUBLISH requests MUST
contain a single SIP-If-Match header field identifying the specific

event state that the request is refreshing, modifying or removing.
This header field MUST contain a single entity-tag that was returned
by the ESC in the SIP-ETag header field of the response to a previous
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.

The presence of a body and the SIP-If-Match header field determine
the specific operation that the request is performing, as described
in Table 1.

```
+-----------+-------+--------------+---------------+
| Operation | Body? | SIP-If-Match? | Expires Value |
+-----------+-------+--------------+---------------+
| Initial   | yes   | no           | > 0           |
| Refresh   | no    | yes          | > 0           |
| Modify    | yes   | yes          | > 0           |
| Remove    | no    | yes          | 0             |
+-----------+-------+--------------+---------------+
```

Table 1: Publication Operations

An 'Initial' publication sets the initial event state for a
particular EPA. A 'Refresh' publication refreshes the lifetime of a
previous publication, whereas a 'Modify' publication modifies the
event state of a previous publication. A 'Remove' publication
requests immediate removal of event state. These operations are
described in more detail in the following sections.

4.2 Creating Initial Publication

An initial publication is a PUBLISH request created by the EPA and
sent to the ESC that establishes soft state for the event package
indicated in the Event header field of the request, and bound to the
address in the Request-URI of the request.

An initial PUBLISH request MUST NOT contain a SIP-If-Match header
field. However, if the EPA expects an appropriate, locally stored
entity-tag to still be valid, it SHOULD first try to modify that
event state as described in Section 4.4, instead of submitting an
initial publication.

An initial PUBLISH request MUST contain a body that contains the
published event state.

An initial PUBLISH request MAY contain a single Expires header field. This value indicates the suggested lifetime of the event state publication.

If an Expires header is not present, the EPA is indicating its desire for the ESC to choose. The ESC may lower the suggested lifetime of the publication, but it will never extend it. The Expires header field in a 2xx response to the initial PUBLISH indicates the actual duration for which the publication will remain active. Unless refreshed before this lifetime is exceeded, the publication will expire.

4.3 Refreshing Event State

An EPA is responsible for refreshing its previously established publications before their expiration interval has elapsed. To refresh a publication, the EPA MUST create a PUBLISH request that includes in a SIP-If-Match header field the entity-tag of the publication to be refreshed.

The SIP-If-Match header field containing an entity-tag conditions the PUBLISH request to refresh a specific event state established by a prior publication. If the entity-tag matches previously published event state at the ESC, the refresh succeeds, and the EPA receives a 2xx response.

Like the 2xx response to an initial PUBLISH request, the 2xx response to a refresh PUBLISH request will contain a SIP-ETag header field with an entity-tag. The EPA MUST store this entity-tag, replacing any existing entity-tag for the refreshed event state. See Section 8.2 for more information on the EPA handling of entity-tags.

If there is no matching event state, e.g., the event state to be refreshed has already expired, the EPA receives a 412 (Conditional Request Failed) response to the PUBLISH request.

A publication refresh MAY contain a single Expires header field. This value indicates the suggested lifetime of the event state.

If an Expires header is not present, the EPA is indicating its desire for the ESC to choose. The ESC may lower the suggested lifetime of the publication refresh, but it will never extend it. The Expires header field in a 2xx response to the publication refresh indicates the actual duration for which the publication will remain active.

A publication refresh only extends the expiration time of already existing event state. It does not affect that event state in any other way. Therefore, a PUBLISH request that refreshes event state

MUST NOT have a body.

4.4 Modifying Event State

Modifying event state closely resembles the creation of initial event
state. However, instead of establishing completely new event state
at the ESC, already existing event state is updated with modified
event state. The nature of this update depends on the content of the
body, and the semantics associated with the format of that body.

To modify event state, the EPA MUST construct a PUBLISH request that
includes in a SIP-If-Match header field the entity-tag of the event
state publication to be modified. A PUBLISH request that modifies
event state MUST contain a body that includes the modified event
state.

The SIP-If-Match header field conditions the PUBLISH request to
modify a specific event state established by a prior publication, and
identified by the entity-tag. If the entity-tag matches previously
published event state at the ESC, that event state is replaced by the
event state carried in the PUBLISH request, and the EPA receives a
2xx response.

Like the 2xx response to an initial PUBLISH request, the 2xx response
to a modifying PUBLISH request will contain a SIP-ETag header field
with an entity-tag. The EPA MUST store this entity-tag, replacing
any existing entity-tag for the modified event state. See Section
8.2 for more information on the EPA handling of entity-tags.

If there is no matching event state at the ESC, e.g., the event state
to be modified has already expired, the EPA receives a 412
(Conditional Request Failed) response to the PUBLISH request.

A modifying PUBLISH request MAY contain a single Expires header
field. This value indicates the suggested lifetime of the event
state publication.

If an Expires header is not present, the EPA is indicating its desire
for the ESC to choose. The ESC may lower the suggested lifetime of
the publication, but it will never extend it. The Expires header
field in a 2xx response to the modifying PUBLISH indicates the actual
duration for which the publication will remain active. Unless
refreshed before this lifetime is exceeded, the publication will
expire.

4.5 Removing Event State

Event state established by a prior publication may also be explicitly

removed.

To request the immediate removal of event state, an EPA MUST create a
PUBLISH request with an Expires value of "0", and set the
SIP-If-Match header field to contain the entity-tag of the event
state publication 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.

Similar to an event state refresh, the removal of event state only
affects the expiry of the event state. Therefore, a PUBLISH request
that removes event state MUST NOT contain a body.

5. Processing PUBLISH Responses

When processing responses to PUBLISH requests, the steps in Section
8.1.2 of RFC 3261 [4] apply.

If an EPA receives a 412 (Conditional Request 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 SIP-If-Match header field, as described in Section
4.2. The EPA MUST also discard the entity-tag that produced this
error resoponse.

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 User Agent Server (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. The ESC MUST ignore the
Contact header field if one is present in a PUBLISH request.

PUBLISH requests with the same Request-URI 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 the steps defining
general UAS behavior in Section 8.2 of RFC 3261 [4]. In addition,
for PUBLISH specific behavior 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.

       It may also be that the Request-URI points to a domain that
       the ESC is not responsible for. In that case, the UAS
       receiving the request can assume the role of a proxy server
       and forward the request to a more appropriate target.

2. 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.

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

    * If the request does not contain a SIP-If-Match header field,
       the ESC MUST generate and store a locally unique entity-tag
       for identifying the publication. This entity-tag is
       associated with the event-state carried in the body of the
       PUBLISH request.

    * Else, if the request has a SIP-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
       SIP-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 (Conditional Request
       Failed), and skip the remaining steps.

4. 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.

5. The ESC processes the published event state contained in the body
    of the PUBLISH request. If 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, such as
    415 (Unsupported Media Type), and skip the remainder of the
    steps.

    * If present, the ESC stores the event state delivered in the
      PUBLISH request and identified by the associated entity-tag,
      updating any existing event state for that 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 an appropriate error
    response, such as 504 (Server Time-out), and skip the last step.

6. The ESC returns a 200 (OK) response. The response MUST contain
    an Expires header field indicating the expiration interval chosen
    by the ESC. The response MUST also contain a SIP-ETag header
    field that contains a single entity-tag identifying the
    publication. The ESC MUST generate a new entity-tag for each
    successful publication, replacing any previous entity-tag
    associated with that event state. See Section 8.3 for more
    information on the ESC handling of entity-tags.

7. **Processing OPTIONS Requests**

   A client may probe the ESC for the support of PUBLISH using the
   OPTIONS request defined in SIP [4]. The ESC processes OPTIONS
   requests as defined in Section 11.2 of RFC 3261 [4]. In the response
   to 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 Allow header field may also be used to specifically announce
      support for PUBLISH messages when registering. (See SIP
      Capabilities [12] for details).

8. **Use of Entity-tags in PUBLISH**

   This section makes a general overview of the entity-tags usage in
   PUBLISH. It is informative in nature and thus contains no normative
   protocol description.

8.1 **General Notes**

   The PUBLISH mechanism makes use of entity-tags, as defined in HTTP/
   1.1 [13]. While the main functionality is preserved, the syntax and
   semantics for entity-tags and the corresponding header fields is
   adapted specifically for use with the PUBLISH method. The main
   differences are:

   o The syntax for entity-tags is a token instead of quoted-string.
      There is also no prefix defined for indicating a weak entity-tag.

   o A PUBLISH precondition can only apply to a single entity-tag, so
      request preconditions with multiple entity-tags are not allowed.

   o A request precondition can't apply to "any" entity, namely there
      is no special "*" entity-tag value defined for PUBLISH.

   o Whereas in HTTP/1.1 returning an entity-tag is optional for origin
      servers, in PUBLISH ESCs are required to always return an
      entity-tag for a successful publication.

   The main motivation for the above adaptation is that PUBLISH is
   conceptually an HTTP PUT, for which only a subset of the features in
   cache validation using entity-tags is allowed in HTTP/1.1. It makes
   little sense to enable features other than this subset for event
   state publication.

   To make it apparent that the entity-tags usage in PUBLISH is similar

but not identical to HTTP/1.1, we have not adopted the header field
names directly from HTTP/1.1, but rather have created similar but
distinct names, as can be seen in Section 11.

8.2 Client Usage

Each successful publication will get assigned an entity-tag which is
then delivered to the EPA in the response to the PUBLISH request.
The EPA needs to store that entity-tag, replacing any previous
entity-tag for that event state. If a request fails with a 412
(Conditional Request Failed) response, the EPA discards the
entity-tag that caused the failure.

Entity-tags are opaque tokens to the EPA. The EPA cannot infer any
further semantics from an entity-tag beyond a simple identifier, or
assume a specific formatting. An entity-tag may be a monotonically
increasing counter, but it may also be a totally random token. It is
up to the ESC implementation as to what the formatting of an
entity-tag is.

8.3 Server Usage

Entity-tags are generated and maintained by the ESC. They are part
of the state maintained by the ESC that also includes the actual
event state and its remaining expiration interval. An entity-tag is
generated and stored for each successful event state publication, and
returned to the EPA in a 200 (OK) response. Each event state
publication from the EPA that updates a previous publication will
include an entity-tag that the ESC can use as a search key in the set
of active publications.

The way in which an entity-tag is generated is an implementation
decision. One possible way to generate an entity-tag is to implement
it as an integer counter that is incremented by one for each
successfully processed publication. Other, equally valid ways for
generating entity-tags exist, and this document makes no
recommendations or preference for a single way.

9. Controlling the Rate of Publication

As the aggregator of state information from potentially many sources,
the ESC can be subject to considerable amounts of publication
traffic. There are ways to reduce the amount of PUBLISH requests
that the ESC receives:

o Choice of the expiration interval for a publication can be
   affected by the ESC. It can insist that an EPA chooses a longer
   expiration value to what it suggests, in case the ESC's local

default minimum expiration value is not reached. Maintaining a
longer default minimum expiration value at the ESC reduces the
rate at which publications are refreshed.

o   Another way of reducing publication traffic is to use a SIP-level
    push-back to quench a specific source of publication traffic. To
    push back on publications from a particular source, the ESC MAY
    respond to a PUBLISH request with a 503 (Service Unavailable), as
    defined in RFC 3261 [4]. This response SHOULD contain a
    Retry-After header field indicating the time interval that the
    publication source is required to wait until sending another
    PUBLISH request.

At the time of writing this specification, work on managing load in
SIP is starting, which may be able to provide further tools for
managing load in event state publication systems.

10. 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.

Any future event package specification SHOULD include a discussion of
its considerations for using PUBLISH. At a minimum those
considerations SHOULD address the issues presented in this chapter,
and MAY include additional considerations.

10.1 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 type, 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 [6] 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.

10.2 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.

10.3 Multiple Sources for Event State

For some event packages, the underlying model is that of a single
aggregator of event state (ESC), and multiple sources, out of which
only some may be using the PUBLISH mechanism.

> Note that sources for event state other than those using the
> PUBLISH mechanism are explicitly allowed. However, it is beyond
> the scope of this document to define such interfaces.

Event packages that make use of the PUBLISH mechanism SHOULD describe
whether this model for event state publication is applicable, and MAY
describe specific mechanisms used for aggregating publications from
multiple sources.

For presence, 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 and out of
the scope of this specification.

10.4 Event State Segmentation

For some event packages, there exists a natural decomposition of
event state into segments. Each segment is defined as one of
potentially many identifiable sections in the published event state.
Any event package whose content type supports such segmentation of
event state, SHOULD describe the way in which these event state
segments are identified by the ESC.

In presence publication, the EPA MUST keep the "id" attributes of
tuples consistent in the context of an entity-tag. If a publication
modifies the contents of a tuple, that tuple MUST maintain its
original "id". The ESC will interpret each tuple in the context of
the entity-tag with which the request arrived. A tuple whose "id" is
missing compared to the original publication will be considered as
being removed. Similarly, a tuple is interpreted as being added if
its "id" attribute is one that the original publication did not
contain.

10.5 Rate of Publication

Controlling the rate of publication is discussed in Section 9.
Individual event packages MAY in turn define recommendations (SHOULD
or MUST strength) on absolute maximum rates at which publications are
allowed to be generated by a single EPA.

There are no rate limiting recommendations for presence publication.

11. Protocol Element Definitions

This section describes the extensions required for event publication
in SIP.

11.1 New Methods

11.1.1 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 compositing this event state.

Table 2 and Table 3 extend Tables 2 and 3 of RFC 3261 [4] by adding
an additional column, defining the header fields that can be used in
PUBLISH requests and responses. The keys in these tables are
specified in Section 20 of RFC 3261 [4].

Header Field	where	PUBLISH
Accept	R	o
Accept	2xx	-
Accept	415	m*
Accept-Encoding	R	o
Accept-Encoding	2xx	-
Accept-Encoding	415	m*
Accept-Language	R	o
Accept-Language	2xx	-
Accept-Language	415	m*
Alert-Info		-
Allow	R	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 2: 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 3: Summary of header fields, P--Z

11.2 New Response Codes

11.2.1 "412 Conditional Request Failed" Response Code

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

11.3 New Header Fields

   Table 4, Table 5, and Table 6 expand on Table 3 in SIP [4], as
   amended by the changes in Section 11.1.

```
+--------------+-------+-------+-----+-----+-----+-----+-----+
| Header Field | where | proxy | ACK | BYE | CAN | INF | INV |
+--------------+-------+-------+-----+-----+-----+-----+-----+
| SIP-ETag     | 2xx   |       |  -  |  -  |  -  |  -  |  -  |
| SIP-If-Match |  R    |       |  -  |  -  |  -  |  -  |  -  |
+--------------+-------+-------+-----+-----+-----+-----+-----+
```

Table 4: Summary of header fields, P--Z

```
+--------------+-------+-------+-----+-----+-----+-----+-----+
| Header Field | where | proxy | NOT | OPT | PRA | REG | SUB |
+--------------+-------+-------+-----+-----+-----+-----+-----+
| SIP-ETag     | 2xx   |       |  -  |  -  |  -  |  -  |  -  |
| SIP-If-Match |  R    |       |  -  |  -  |  -  |  -  |  -  |
+--------------+-------+-------+-----+-----+-----+-----+-----+
```

Table 5: Summary of header fields, P--Z

```
+--------------+-------+-------+-----+-----+-----+---------+
| Header Field | where | proxy | UPD | MSG | REF | PUBLISH |
+--------------+-------+-------+-----+-----+-----+---------+
| SIP-ETag     | 2xx   |       |  -  |  -  |  -  |    m    |
| SIP-If-Match |  R    |       |  -  |  -  |  -  |    o    |
+--------------+-------+-------+-----+-----+-----+---------+
```

Table 6: Summary of header fields, P--Z

11.3.1 "SIP-ETag" Header Field

SIP-ETag is added to the definition of the element "general-header"
in the SIP message grammar. Usage of this header is described in
Section 4 and Section 6.

11.3.2 "SIP-If-Match" Header Field

SIP-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 4 and Section 6.

12. Augmented BNF Definitions

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 [7] format used in SIP
[4], and contain references to elements defined therein.

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

13. IANA Considerations

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

13.1 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).

13.2 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:  Conditional Request Failed
```

13.3 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:   SIP-ETag
      Compact Form:  (none)

      Header Name:    SIP-If-Match
```

Compact Form: (none)

14. Security Considerations

14.1 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.

ESCs and EPAs MUST implement Digest for authenticating PUBLISH
requests, as defined in RFC 3261 [4]. The exact methods for creating
and manipulating access control policies in the ESC are outside the
scope of this document.

14.2 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. Implementations
MUST support Digest authentication, as defined in RFC 3261 [4].

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 minimum 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.

14.3 Replay Attack

Replaying a 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
that 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 MUST support Digest
authentication with replay protection, as defined in RFC 3261 [4].
Further mechanisms for countering replay attacks are discussed in SIP
[4].

14.4 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,
SIP-If-Match, Route, and Expires headers and the bodies of PUBLISH
requests.

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.

For integrity protection, ESCs MUST implement TLS [8], and MUST
support both mutual and one-way authentication, and MUST also support
the SIPS URI scheme defined in SIP [4]. EPAs SHOULD be capable of
initiating TLS and SHOULD support the SIPS URI scheme. ESCs and EPAs
MAY support S/MIME [9] for integrity protection, as defined in SIP
[4].

14.5 Confidentiality

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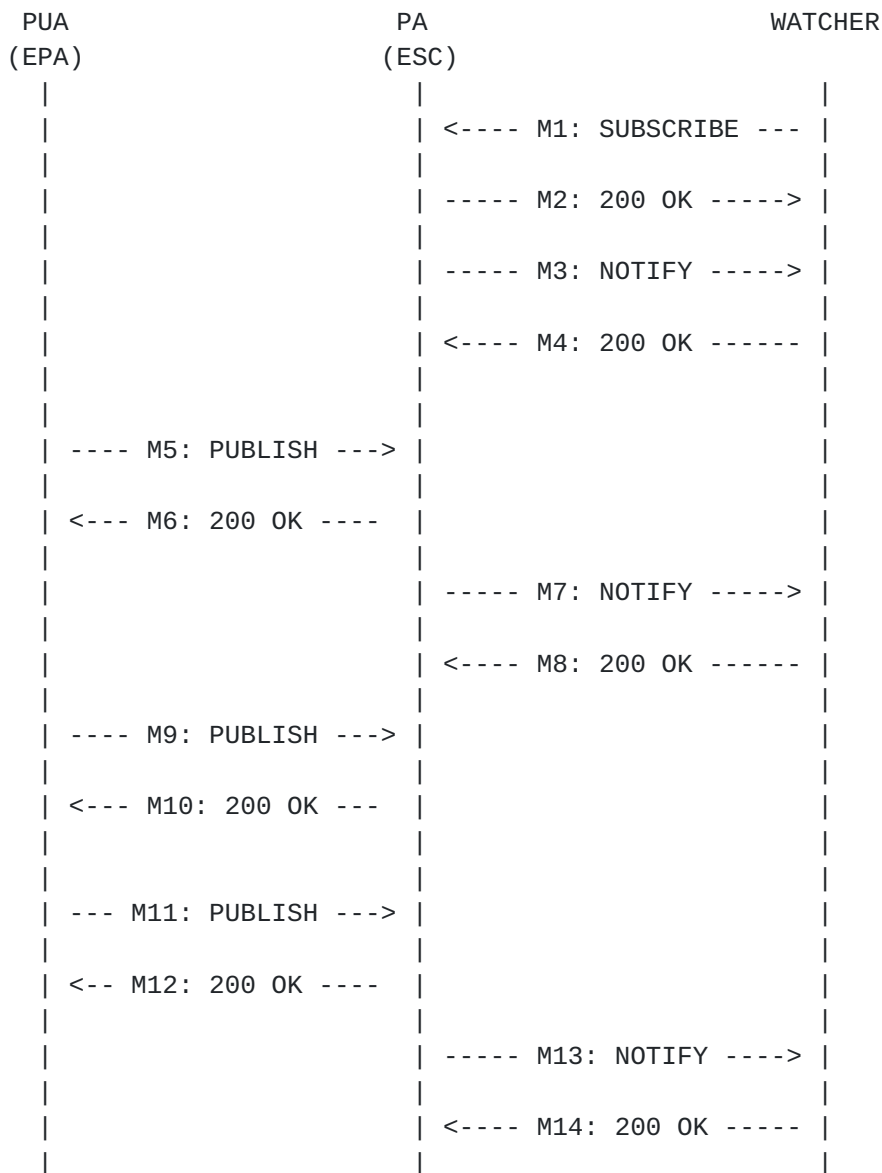information to ensure confidentiality.

For providing confidentiality, ESCs MUST implement TLS [8], MUST
support both mutual and one-way authentication, and MUST also support
the SIPS URI scheme defined in SIP [4]. EPAs SHOULD be capable of
initiating TLS and SHOULD support the SIPS URI scheme. ESCs and EPAs
MAY support S/MIME [9] for encryption of event state information, as
defined in SIP [4].

15. 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.

```
         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 host.example.com;branch=z9hG4bKnashds7
    To: <sip:presentity@example.com>
    From: <sip:watcher@example.com>;tag=12341234
    Call-ID: 12345678@host.example.com
    CSeq: 1 SUBSCRIBE
    Max-Forwards: 70
    Expires: 3600
    Event: presence
    Contact: sip:user@host.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 host.example.com;branch=z9hG4bKnashds7
     ;received=192.0.2.1
    To: <sip:presentity@example.com>;tag=abcd1234
    From: <sip:watcher@example.com>;tag=12341234
    Call-ID: 12345678@host.example.com
    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:user@host.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@host.example.com
      CSeq: 1 NOTIFY
      Max-Forwards: 70
      Event: presence
      Subscription-State: active; expires=3599
      Contact: sip:pa.example.com
      Content-Type: application/pidf+xml
      Content-Length: ...

      <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns: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="gwewg991">
            <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
       ;received=192.0.2.2
      To: <sip:watcher@example.com>;tag=12341234
      From: <sip:presentity@example.com>;tag=abcd1234
      Call-ID: 12345678@host.example.com
      CSeq: 1 NOTIFY
```

   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/pidf+xml
      Content-Length: ...

      <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
                entity="pres:presentity@example.com">
         <tuple id="efeef223">
            <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
    SIP-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
       ;received=192.0.2.3
      To: <sip:presentity@example.com>;tag=1a2b3c4d
      From: <sip:presentity@example.com>;tag=1234wxyz
      Call-ID: 81818181@pua.example.com
      CSeq: 1 PUBLISH
      SIP-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:user@host.example.com SIP/2.0
Via: SIP/2.0/UDP pa.example.com;branch=z9hG4bK4cd42a
To: <sip:watcher@example.com>;tag=12341234
From: <sip:presentity@example.com>;tag=abcd1234
Call-ID: 12345678@host.example.com
CSeq: 2 NOTIFY
Max-Forwards: 70
Event: presence
Subscription-State: active; expires=3400
Contact: sip:pa.example.com
Content-Type: application/pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
          entity="pres:presentity@example.com">
   <tuple id="efeef223">
      <status>
         <basic>closed</basic>
      </status>
      <timestamp>2003-02-01T17:00:19Z</timestamp>
   </tuple>
   <tuple id="gwewg991">
      <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 pa.example.com;branch=z9hG4bK4cd42a
 ;received=192.0.2.2
To: <sip:watcher@example.com>;tag=12341234
From: <sip:presentity@example.com>;tag=abcd1234
Call-ID: 12345678@host.example.com
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
    SIP-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. As always,
     the ESC returns an entity-tag in the response to a successful
     PUBLISH. 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
     ;received=192.0.2.3
    To: <sip:presentity@example.com>;tag=2affde434
    From: <sip:presentity@example.com>;tag=1234kljk
    Call-ID: 98798798@pua.example.com
    CSeq: 1 PUBLISH
    SIP-ETag: kwj449x
    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
      SIP-If-Match: kwj449x
      Expires: 3600
      Event: presence
      Content-Type: application/pidf+xml
      Content-Length: ...

      <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
                entity="pres:presentity@example.com">
         <tuple id="efeef223">
            <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
       ;received=192.0.2.3
      To: <sip:presentity@example.com>;tag=effe22aa
      From: <sip:presentity@example.com>;tag=54321mm
      Call-ID: 5566778@pua.example.com
      CSeq: 1 PUBLISH
      SIP-ETag: qwi982ks
      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:user@host.example.com SIP/2.0
Via: SIP/2.0/UDP pa.example.com;branch=z9hG4bK32defd3
To: <sip:watcher@example.com>;tag=12341234
From: <sip:presentity@example.com>;tag=abcd1234
Call-ID: 12345678@host.example.com
CSeq: 2 NOTIFY
Max-Forwards: 70
Event: presence
Subscription-State: active; expires=3400
Contact: sip:pa.example.com
Content-Type: application/pidf+xml
Content-Length: ...

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
          entity="pres:presentity@example.com">
   <tuple id="efeef223">
      <status>
         <basic>open</basic>
      </status>
      <timestamp>2003-02-01T19:15:15Z</timestamp>
   </tuple>
   <tuple id="gwewg991">
      <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 pa.example.com;branch=z9hG4bK32defd3
 ;received=192.0.2.3
To: <sip:watcher@example.com>;tag=12341234
From: <sip:presentity@example.com>;tag=abcd1234
Call-ID: 12345678@host.example.com
CSeq: 2 NOTIFY
Content-Length: 0
```

16. 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.

17. Acknowledgements

The authors would like to thank the SIMPLE Working Group for their
collective effort, and specifically the following people for their
review and support of this work: Henning Schulzrinne, Paul Kyzivat,
Hisham Khartabil, George Foti, Keith Drage, Samir Srivastava, Arun
Kumar, Adam Roach, Pekka Pessi, Kai Wang, Cullen Jennings, Mikko
Lonnfors, Eva-Maria Leppanen, Ernst Horvath, Thanos Diacakis, Oded
Cnaan, Rohan Mahy and Dean Willis.

18. Document Change History

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

18.1 Changes since "draft-ietf-sip-publish-03"

The following changes were made since the last version:

o Changed the default response phrase for 412 to avoid clashing with
   session preconditions

o Added explanations for columns and keys therein in tables 2 and 3
   for clarity

o Reworded passages in sections 4.3 and 4.4, and added a reference
   to chapter 8 to clarify the etags usage in refresh/modify
   publications.

o Added a reference to section 8 in the last step of section 6 to
   clarify the etags usage by the ESC.

o Small wording change in section 8.2, 1st paragraph to clarify the
   meaning of the 2nd sentence.

18.2 **Changes since "draft-ietf-sip-publish-02"**

The following changes were made since the last version:

o Changed title to be in line with draft-ietf-sip-guidelines.

o Added RFC 2779 reference, as well as a reference for
   draft-ietf-simple-presence.

o Added a definition for presence compositor.

o Cleaned up the message syntax in the examples section.

o Fixed typos and improved wording.

18.3 **Changes since "draft-ietf-sip-publish-01"**

The following changes were made since the last version:

o Added new chapter discussing entity-tags in general.

o Added new chapter discussing rate control for publications,
   including SIP level push-back.

o Added back a considerations section for event segmentation (in
   Chapter 4), and clarified text in other parts.

o Clarified text on constructing a PUBLISH. Added a table
   describing the operations and their properties.

o Changed syntax by adding a "SIP-" prefix to the header field
   names. This is to indicate that the syntax/semantics of
   entity-tags is similar but different from the HTTP counterparts.

o Fixed the draft to consistently use Request-URI as identifying the
   target resource for the publication.

o Clarified Contact usage and in-dialog requests.

o Lots of fixes to various places in the draft based on review
   comments.

o Split the old Table 3 into two for better readability.

o Fixed examples to use correct PIDF XML namespace declarations and
   MIME type.

o Added reference to ABNF.

18.4 **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.

18.5 **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".

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

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

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

o 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.

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

o Added methods other than PUBLISH to Table 3.

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

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

o Overall language and structure tweaking.

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

   The following changes were made since the last version:

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

o Removed usage of Call-ID and CSeq for ordering

o Removed timestamp based versioning

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

o Changed reference to content-indirection as Informative

o 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.

19. References

19.1 Normative References

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

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

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

   [4]   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.

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

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

   [7]   Crocker, D. and P. Overell, "Augmented BNF for Syntax
         Specifications: ABNF", RFC 2234, November 1997.

   [8]   Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC
         2246, January 1999.

   [9]   Ramsdell, B., "S/MIME Version 3 Message Specification", RFC
         2633, June 1999.

19.2 Informative References

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

   [11] Mahy, R., "A Message Summary and Message Waiting Indication
         Event Package for the Session Initiation Protocol (SIP)",
         draft-ietf-sipping-mwi-04 (work in progress), December 2003.

   [12] Rosenberg, J., "Indicating User Agent Capabilities in the
         Session Initiation Protocol (SIP)",
         draft-ietf-sip-callee-caps-03 (work in progress), January 2004.

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

Author's Address

   Aki Niemi (editor)
   Nokia
   P.O. Box 100
   NOKIA GROUP, FIN 00045
   Finland

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

Intellectual Property Statement

Full Copyright Statement

Acknowledgment