Network Working Group
Request for Comments: 5263
Category: Standards Track

M. Lonnfors
J. Costa-Requena
E. Leppanen
Nokia
H. Khartabil
Ericsson

September 2008

Session Initiation Protocol (SIP) Extension for Partial Notification of Presence Information

Status of This Memo

Abstract

By default, presence delivered using the presence event package for the Session Initiation Protocol (SIP) is represented in the Presence Information Data Format (PIDF). A PIDF document contains a set of elements, each representing a different aspect of the presence being reported. When any subset of the elements change, even just a single element, a new document containing the full set of elements is delivered. This memo defines an extension allowing delivery of only the presence data that has actually changed.

Table of Contents

1. Introduction

A presence event package for Session Initiation Protocol (SIP) [3] allows users ('watchers') to subscribe to other users' ('presentities') presence information. The presence information is composed of multiple pieces of data that are delivered to the watcher. The size of the presence information document can be large (i.e., the presence document can contain an arbitrary number of elements called presence tuples that convey data). As specified in RFC 2778 [9] and the presence event package for SIP [3], a Presence Agent (PA) always delivers in presence notifications all the presence data that has been authorized for a certain watcher. This is done regardless of what presence data has changed compared to last notification. It may not be reasonable to send the complete presence information over low bandwidth and high latency links when only part of the presence information has actually changed. This may end up degrading the presence service and causing bad perception at the watcher side.

This document defines a partial notification approach where the presence server delivers to the watchers only those parts of the presence information that have changed compared to the presence information sent in the previous notifications. This reduces the amount of data that is transported over the network.

This mechanism utilizes the presence event package for SIP [3] and a new MIME type for carrying partial Presence Information Data Format documents [2].

2. Conventions

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

This document makes use of the vocabulary defined in RFC 2778 [9], RFC 3265 [6], the presence event package for SIP [3], and the PIDF extension for Partial Presence [2].

3. Introduction to the Partial Notification Mechanism

This chapter briefly introduces the regular functionality of the presence service, and gives an overview of the partial notification solution. This section is informational in nature. It does not contain any normative statements.

3.1. Basic Presence Agent Operation

The presence service normally operates so that a watcher sends a SIP SUBSCRIBE request targeted to the presentity. The request is routed to the presence agent where the presentity's presence information is stored. The SUBSCRIBE request can include an Accept header field that indicates the supported content types.

The presence agent receives the SUBSCRIBE request and if there is no Accept header indicating the supported content types or the Accept header contains the default PIDF content type, the PA will generate presence notifications using the default PIDF format [5]. The PIDF document can contain one or multiple XML elements. The PIDF document includes a set of elements defined in RFC 2778 [9], and its extensions for representing the presence information. This PIDF document will be carried in the body of a NOTIFY request constructed as per RFC 3265 [6]. During basic operation, the presence document always contains the full state corresponding to the presence status of the presentity, as determined by the PA local policy and authorization rules.

3.2. Operation with Partial Notification

The partial notification mechanism allows a watcher to request that, whenever possible, notifications contain only presence information that has actually changed. A watcher that wants to receive partial notifications according to this document, creates a SIP SUBSCRIBE request similar to that of a regular presence subscription. However, the SIP SUBSCRIBE request contains an Accept header field whose value

contains the "application/pidf-diff+xml" tag as well as the "application/pidf+xml" tag.

When the presence agent receives the subscription, it examines the Accept header field value and if the "application/pidf-diff+xml" value is present, it can decide to use the partial notifications mechanism specified in this memo. The presence agent builds NOTIFY requests that contain the Content-Type header field set to "application/pidf-diff+xml". The first NOTIFY request that contains presence information will contain a full presence document. Subsequent NOTIFY requests can contain partial presence documents. This behavior is described in detail in Section 4.

4. Client and Server Operations

Unless otherwise specified in this document, the regular watcher and presence agent behavior is applied as defined in the SIP presence event package [3].

4.1. Content-Type for Partial Notifications

Entities supporting the partial notification extension described in this document MUST support the 'application/pidf-diff+xml' content type specified in the PIDF extension for partial presence [2].

4.2. Watcher Generation of SUBSCRIBE Requests

A SUBSCRIBE request can be used to negotiate the preferred content type to be used in the notifications. The Accept header field is used for this purpose as specified in RFC 3261 [4]. When a watcher wants to allow the presence agent to send partial notifications the watcher MUST include an Accept header field in its SUBSCRIBE request. The value of the Accept header field MUST contain 'application/pidf-diff+xml' (in addition to 'application/pidf+xml' required by the SIP presence event package [3]). The watcher MAY include a "q" parameter with each Accept value to indicate the relative preference of that value.

4.3. Presence Agent Processing of SUBSCRIBE Requests

The presence agent receives subscriptions from watchers and generates notifications according to the SIP presence event package [3]. If the watcher has indicated the supported content types in the Accept header field of the SUBSCRIBE request, the presence agent compares the values included in the Accept header field with the supported ones, and decides which one to use. If the watcher has indicated preferred accept values by means of "q" parameters, the presence

agent SHOULD base the decision on those preferences, unless otherwise indicated by the presence agent local policy.

4.4. Presence Agent Generation of Partial Notifications

Once a subscription is accepted and installed, the PA MUST deliver the full state of the presence information in the first partial notification that contains a presence document having <pidf-full> root element. If the presence agent decides to send notifications that include a presence document according to this specification, the presence agent MUST build a presence document according to the PIDF extension for Partial Presence [2] and MUST set the Content-Type header field to the value 'application/pidf-diff+xml'.

When using the 'application/pidf-diff+xml' MIME type, the PA MUST include a "version" attribute; for the first partial notification (within a given subscription), the PA MUST initialize version to value one (1). This version counter is scoped to the subscription, and is incremented by one within each partial notification. The version value is only reset when the given subscription is terminated. It is not reset when the subscription is refreshed.

When the PA generates a partial presence document, the PA SHOULD include only that presence information that has changed compared to the previous notifications. It is up to the PA's local policy to determine what is considered as a change to the previous state.

The PA MUST construct the partial presence document according to the following logic:

- The PA MUST construct the presence information according to the PIDF extension for Partial Presence [2]. All the information that has been added to the presence document is listed inside <add> elements. All information that has been removed from the presence document is listed inside <remove> elements, and all information that has been changed is listed under <replace> elements.

- The PA MUST include a "version" attribute in the presence document. The PA MUST increment the version number by one compared to the earlier successfully sent presence document in the PIDF extension for Partial Presence [2] format to the watcher associated with a certain subscription.

The PA MUST NOT send a new NOTIFY request that contains a partial notification for the same Request-URI until it has received a final response from the watcher for the previous one or the previous NOTIFY request has timed out.

When the PA receives a SUBSCRIBE request (refresh or termination) within the associated subscription, it SHOULD send a NOTIFY request containing the full presence document.

If the PA has used other than the 'application/pidf-diff+xml' content type in notifications within the existing subscription, and changes to deliver partial notifications, the PA MUST deliver the full state of the presence information containing a presence document having <pidf-full> root element as the first partial notification.

4.5. Watcher Processing of NOTIFY Requests

Watcher processes all NOTIFY requests that contain 'application/pidf+xml' content type as specified in RFC 3856 [3].

When the watcher receives the first notification containing the 'application/pidf-diff+xml' MIME body the watcher MUST initialize an internal version counter, related to this subscription, to the value of the "version" included in the presence document. This version counter is scoped to the subscription. The watcher MUST also store the received full presence document as its local copy.

When the watcher receives a subsequent 'application/pidf-diff+xml' encoded presence document the watcher MUST compare the received "version" attribute with the local version counter. If the watcher receives a presence document with the "version" attribute value equal or lower than the locally stored version number, it is considered a PA failure, and the watcher SHOULD discard the document without further processing. Otherwise, the watcher MUST modify its locally stored information according to the following logic:

- If the root element of the presence document is <pidf-full>, the watcher must replace its local copy of the presence document with the presence document received in the 'application/pidf-diff+xml' body and set the internal version value to the value of the "version" attribute included in the presence document.

- If the root element of the presence document is <pidf-diff> and the received version number is incremented by one compared with the local version counter, the watcher MUST apply the changes to its local copy of the full presence document indicated in the received 'application/pidf-diff+xml' document as specified in PIDF extension for Partial Presence [2]. The watcher MUST increment the local version counter by one.

- If the root element of the presence document is <pidf-diff> and the received "version" value is higher by more than one compared with the locally stored value, the watcher assumes that one or more NOTIFYs were lost. The watcher SHOULD either refresh the subscription in order to receive a full presence document or terminate the subscription.

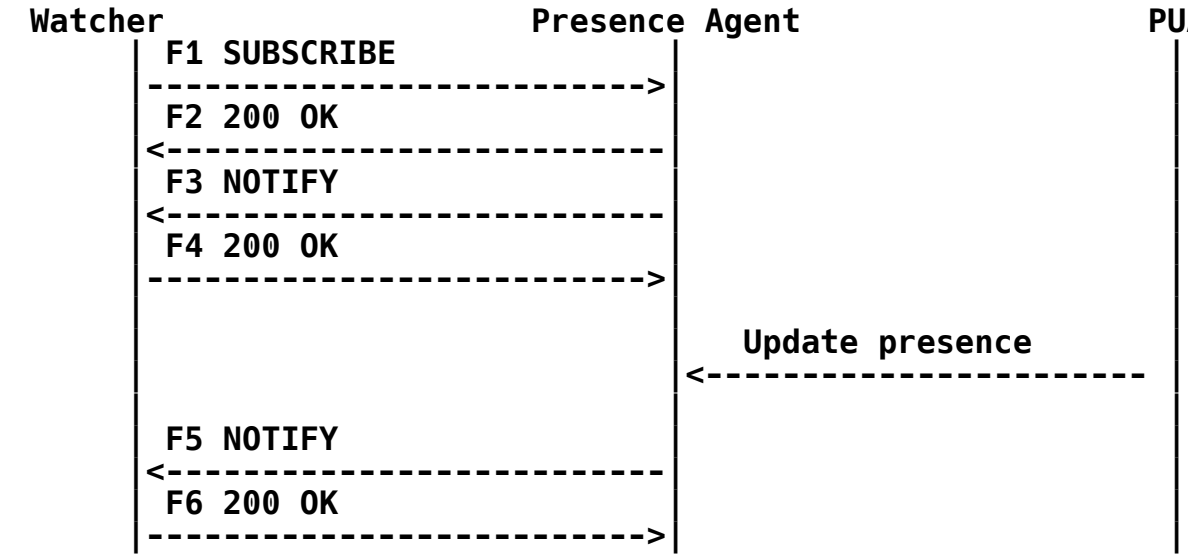
If the watcher encounters a processing error while processing the received 'application/pidf-diff+xml' encoded presence document, look at Section 5.1 of [8]. In this case, the watcher SHOULD renew the subscription. The watcher MAY also fall back to normal presence operations by not inserting 'application/pidf-diff+xml' in a new SUBSCRIBE request. It is hardly reasonable to signal this error to the notifier even if the error exists in the notifier process.

If the PA changes the content type used in notifications within the existing subscription, the watcher MUST discard all the previously received presence information (except local version counter) from that particular presentity and process the new content as specified for that content type. The local version counter MUST NOT be discarded because if the PA changes back to 'application/ pidf-diff+xml', the MIME type version counter will continue to increase from the last version value.

5. Examples

The following message flow shows an example applying the partial notifications mechanism.

A watcher sends a SUBSCRIBE request declaring support for the default presence format ('application/pidf+xml) and for the partial notification format ('application/pidf-diff+xml') in the Accept header field value. The watcher uses the "q" parameter to set the preference for receiving partial notifications. The PA accepts the subscription and, based on the "q" parameter value, selects to send partial notifications in NOTIFY requests. The first NOTIFY request includes the full state of presence information. The following notifications only include information about delta of the presence information from the previous NOTIFY requests.



Message Details

F1 SUBSCRIBE watcher->example.com server

```
SUBSCRIBE sip:resource@example.com SIP/2.0
Via: SIP/2.0/TCP watcherhost.example.com;
  branch=z9hG4bKnashds7
To: <sip:resource@example.com>
From: <sip:watcher@example.com> ;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Max-Forwards: 70
Event: presence
Accept: application/pidf+xml;q=0.3,
  application/pidf-diff+xml;q=1
Contact: <sip:user@watcherhost.example.com>
Expires: 3600
Content-Length: 0
```

The PA accepts the subscription and generates a 200 OK response to the SUBSCRIBE request

F2 200 OK example.com server ->watcher

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP watcherhost.example.com;
  branch=z9hG4bKnashds7
  ;received=192.0.2.1
To: <sip:resource@example.com>;tag=ffd2
From: <sip:watcher@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Event: presence
Expires: 3600
Contact: <sip:server.example.com>
Content-Length: 0
```

The PA, based on the "q" parameter value in the Accept header of the SUBSCRIBE request (F1), decides to use partial notifications. The PA creates the first NOTIFY request that includes the full presence document.

F3 NOTIFY example.com server -> watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/TCP server.example.com;
  branch=z9hG4bKna998sk
To: <sip:watcher@example.com>;tag=xfg9
From: <sip:resource@example.com>;tag=ffd2
Call-ID: 2010@watcherhost.example.com
Event: presence
Subscription-State: active;expires=3599
Max-Forwards: 70
CSeq: 8775 NOTIFY
Contact: <sip:server.example.com>
Content-Type: application/pidf-diff+xml
Content-Length: ...
```

```
<?xml version="1.0" encoding="UTF-8"?>
  <p:pidf-full xmlns="urn:ietf:params:xml:ns:pidf"
        xmlns:p="urn:ietf:params:xml:ns:pidf-diff"
        xmlns:r="urn:ietf:params:xml:ns:pidf:rpid"
        xmlns:c="urn:ietf:params:xml:ns:pidf:caps"
        xmlns:cp="urn:ietf:params:xml:ns:pidf:cipid"
        xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
        entity="sip:resource@example.com"
        version="1">
```

```
<tuple id="sg89ae">
 <status>
  <basic>open</basic>
 </status>
 <c:servcaps>
  <c:audio>true</c:audio>
  <c:video>false</c:video>
  <c:message>true</c:message>
 </c:servcaps>
  <r:relationship><r:assistant/></r:relationship>
 <contact priority="0.8">tel:09012345678</contact>
</tuple>

<tuple id="cg231jcr">
 <status>
  <basic>open</basic>
 </status>
 <contact priority="1.0">im:res@example.com</contact>
</tuple>

<tuple id="r1230d">
 <status>
  <basic>closed</basic>
 </status>
 <cp:homepage>http://example.com/~res/</cp:homepage>
 <cp:icon>http://example.com/~res/icon.gif</cp:icon>
 <cp:card>http://example.com/~res/card.vcd</cp:card>
 <contact priority="0.9">sip:resource@example.com</contact>
</tuple>

<note xml:lang="en">Full state presence document</note>

<dm:person id="fdkfj">
  <r:activities>
   <r:on-the-phone/>
   <r:busy/>
  </r:activities>
</dm:person>
```

```
       <dm:device id="u00b40c7">
         <c:devcaps>
          <c:mobility>
           <c:supported>
            <c:mobile/>
           </c:supported>
          </c:mobility>
         </c:devcaps>
         <dm:deviceID>mac:xxx</dm:deviceID>
       </dm:device>

     </p:pidf-full>
```

F4 200 OK watcher -> example.com server

```
      SIP/2.0 200 OK
      Via: SIP/2.0/TCP server.example.com;
        branch=z9hG4bKna998sk
        ;received=192.0.2.2
      To: <sip:watcher@example.com>;tag=xfg9
      From: <sip:resource@example.com>;tag=ffd2
      Call-ID: 2010@watcherhost.example.com
      CSeq: 8775 NOTIFY
      Content-Length: 0

      At a later time, the presentity's presence information
      changes.  The PA generates a NOTIFY request
      that includes information about the changes.
```

F5 NOTIFY example.com server -> watcher

```
      NOTIFY sip:user@watcherhost.example.com SIP/2.0
      Via: SIP/2.0/TCP server.example.com;
        branch=z9hG4bKna998sl
      To: <sip:watcher@example.com>;tag=xfg9
      From: <sip:resource@example.com>;tag=ffd2
      Call-ID: 2010@watcherhost.example.com
      CSeq: 8776 NOTIFY
      Event: presence
      Subscription-State: active;expires=3543
      Max-Forwards: 70
      Contact: <sip:server.example.com>
      Content-Type: application/pidf-diff+xml
      Content-Length: ...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<p:pidf-diff xmlns="urn:ietf:params:xml:ns:pidf"
             xmlns:p="urn:ietf:params:xml:ns:pidf-diff"
             xmlns:r="urn:ietf:params:xml:ns:pidf:rpid"
             xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
          entity="sip:resource@example.com"
          version="2">

 <p:add sel="presence/note" pos="before"><tuple id="ert4773">
  <status>
   <basic>open</basic>
  </status>
  <contact priority="0.4">mailto:res@example.com</contact>
  <note xml:lang="en">This is a new tuple inserted
        between the last tuple and note element</note>
 </tuple>

 </p:add>

 <p:replace sel="*/tuple[@id='r1230d']/status/basic/text()"
  >open</p:replace>

 <p:remove sel="*/dm:person/r:activities/r:busy"/>

 <p:replace sel="*/tuple[@id='cg231jcr']/contact/@priority"
  >0.7</p:replace>

</p:pidf-diff>
```

F6 200 OK watcher-> example.com server

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

6. Security Considerations

This specification relies on the presence event package for SIP [3]. Partial notifications can reveal information about what has changed compared to the previous notification. This can make it easier for an eavesdropper to know what kind of changes are happening in the presentity's presence information. However, the same information can be found if the presence event package is used with baseline PIDF [5].

A third party can inject a NOTIFY request with partial state that will cause the watcher to think it has missed a partial notification and to request a new full presence document. This is no worse than what we have without this extension since a party that could perform such action could also send a NOTIFY with Subscription-State: terminated and achieve the same effect without knowing about the extension. Partial Notification does not make the situation any worse, and the protection mechanisms from the existing system apply to preventing this attack against the partial notification mechanism.

Presence-related security considerations are extensively discussed in the presence event package for SIP [3] and all those identified security considerations apply to this document as well. Issues described in the presence event package for SIP [3], including confidentiality, message integrity and authenticity, outbound authentication, replay prevention, Denial-of-Service (DoS) attacks against thirst parties and DoS attacks against servers all apply here without any change.

It is RECOMMENDED that TLS [7] be used between elements to provide hop-by-hop confidentially protection. Furthermore, S/MIME MAY be used for integrity and authenticity of SUBSCRIBE and NOTIFY requests. This is described in Section 23 of RFC 3261.

7. Acknowledgments

The authors would like to thank Jari Urpalainen, Jyrki Aarnos, Jonathan Rosenberg, Dean Willis, Kriztian Kiss, Juha Kalliokulju, Miguel Garcia, Anders Kristensen, Yannis Pavlidis, Ben Cambell, Robert Sparks, and Tim Moran for their valuable comments.

8. References

8.1. Normative References

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

[2] Lonnfors, M., Khartabil, H., Leppanen, E., and J. Urpalainen, "Presence Information Data Format (PIDF) Extension for Partial Presence", RFC 5262, August 008.

[3] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[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] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

[6] Roach, A., "SIP-Specific Event Notification", RFC 3265, June 2002.

[7] Dierks, T. and E. Rescorla, "The TLS Protocol Version 1.1", RFC 4346, April 2006.

[8] Urpalainen, J., "An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors", RFC 5261, September 2008.

8.2. Informative References

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

Authors' Addresses

Mikko Lonnfors
Nokia
P.O. Box 321
Helsinki
Finland

Phone: +358 71 8008000
EMail: mikko.lonnfors@nokia.com

Jose Costa-Requena
Nokia
P.O. Box 321
Helsinki
Finland

Phone: +358 71 8008000
EMail: jose.costa-requena@nokia.com

Eva Leppanen
Nokia
Lempaala
Finland

EMail: eva.leppanen@saunalahti.fi

Hisham Khartabil
Ericsson
Melbourne
Australia

Phone: +61 416 108 890
EMail: hisham.khartabil@gmail.com

Full Copyright Statement

Intellectual Property