Session Initiation Protocol (SIP) extension for Partial Notification of Presence Information
draft-ietf-simple-partial-notify-02

Abstract

A Presence service can have constraints for delivering presence information to devices with low data processing capabilities, small display, and limited battery power. Limitations can also be caused by the interface between the terminal and the network, i.e. radio links with high latency and low bandwidth. This memo presents a solution that aids in reducing the impact of those constrains and to increase transport efficiency, by introducing a mechanism called partial
notification.

Table of Contents

1. Introduction ............................................. 3
2. Conventions ............................................. 3
3. Introduction to the partial notification mechanism .......... 4
   3.1 Basic presence agent operation ...................... 4
   3.2 Operation with partial notification ................. 4
4. Client and server operations ............................... 5
   4.1 Content-type for partial notifications ............... 5
   4.2 Watcher generation of SUBSCRIBE requests ............ 5
   4.3 Presence agent processing of SUBSCRIBE requests ...... 6
   4.4 Presence agent generation of partial notifications ... 6
   4.5 Watcher processing of NOTIFY requests ............... 7
5. Examples ................................................ 9
6. Security Considerations ................................... 13
   6.1 Confidentiality ....................................... 13
   6.2 Message Integrity and Authenticity ................... 13
   6.3 Outbound Authentication ................................ 13
   6.4 Replay Prevention ..................................... 14
   6.5 Denial of Service Attacks Against Third Parties ....... 14
   6.6 Denial Of Service Attacks Against Servers .......... 14
7. Acknowledgements ......................................... 14
8. References ................................................ 14
   8.1 Normative references .................................. 14
   8.2 Informative references ................................ 15
     Authors' Addresses ....................................... 15
     Intellectual Property and Copyright Statements .......... 17
1. Introduction

A presence event package for the Session Initiation Protocol (SIP) [4] allow 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 RFC2778 [3] and presence event package for the SIP [4] 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.

There are some mechanisms, such as signaling compression (SigComp) [10] and content indirection [9] that can be used to help in this problem. However these solutions set additional requirements on basic network functionalities such as security. Some of the existing solutions enforce certain requirements on the network and terminals for supporting compression mechanism, while other solutions require having a specific server to store the requested presence information until the terminal fetches it using another protocol (e.g. HTTP) and, therefore, increases possible security concerns.

This draft presents a solution to problems described above, called partial notifications.

In general, the partial notification approach means that 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 needs be transported over the network.

This mechanism utilizes the presence event package for SIP [4] and the partial PIDF MIME type [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 RFC2778 [3], RFC3265 [7], presence event package for the SIP [4], and partial PIDF format [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 information in nature and it does not provide 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 [6]. The PIDF document can contain one or multiple tuples and presence document level information. The tuples include a set of elements defined in RFC2778 [3] for representing the presence information. NOTIFY requests, built according to RFC 3265 [7], carry in the body of the request a presence document containing presence information. Unless otherwise specified, the presence document 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 enables the watcher to request those parts of the presence information that have changed since the last notifications was sent. The presence agent will send only the changed presence information since the last notification.

A watcher that wants to receive partial notifications according to this specification, 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-partial+xml" tag. Besides the Accept header field, there is no other difference between a regular presence subscription and a subscription requesting partial notifications.

When the presence agent receives the subscription, it examines the
Accept header field value and decides to use the partial notifications mechanism specified in this memo. The presence agent builds NOTIFY requests that contain the Accept header field set to "application/pidf-partial+xml". The first NOTIFY request that contains presence information will contain a full presence document. This is identified by the "state" attribute set to "full" in the XML presence document. Subsequent NOTIFY requests will contain partial notifications, identified by the "state" attribute set to "partial" in the XML presence document.

This means that if a partial notification contains new tuples (tuples which have new tuple ids compared to the full presence document) they are added to the local full presence document. If it contains tuples which have existing tuple ids it means that those tuples are updated. If "removed" element in the XML document contains existing tuple ids it means that those tuples are removed. The watcher updates the local copy accordingly. This behavior is described in detail in Section 4.

Partial notifications, within the scope of this document, apply only to <tuple> level XML elements and all possible elements that are contained in these elements, i.e., tuples are considered to be atomic data elements. This means that, if a notification contains an update to an existing tuple, the updated tuple replaces the existing one. The SIP presence event package [4] provides instructions to process all the data elements which are located outside <tuple> elements. This typically means that every notification will include all those XML elements (for example the <note> element).

4. Client and server operations

This document assumes, that unless otherwise specified in this document, the regular watcheer and presence agent behavior is applied as defined in the SIP presence event package [4].

4.1 Content-type for partial notifications

Entities supporting the partial notification extension described in this document MUST support the 'application/pidf-partial+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 RFC2161 [5]. When a watcher wants to allow the presence agent to send partial notifications the watcher MUST include an Accept header field in a SUBSCRIBE request. The value
of the Accept header file MUST contain ‘application/pidf-partial+xml’ (in addition to ‘application/pidf+xml’ required by the SIP presence event package [4]). The watcher MAY include a "q" parameter to each accept value to indicate the weight of the accept value, i.e., the most preferred accept value.

4.3 Presence agent processing of SUBSCRIBE requests

The presence agent receives the subscriptions from watchers and generates the notifications according to SIP presence event package [4]. 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

If the presence agent decides to send notifications according to this specification that include a presence document, 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-partial+xml'.

Tuple ids are used to match tuples across subsequent notifications. Presence agent MUST use the same tuple ids for tuples which are identical between subsequent notifications and MUST allocate different tuple ids for tuples which are different from previously sent tuples. Presence agents MUST keep tuple ids consistent until the next full state presence document is delivered. The decision on whether tuple ids are the same or different is left up to PA's local policy.

Once a subscription is accepted and installed, the PA MUST deliver the full state of the presence information in the first notification that contains a presence document. In this case, the PA MUST set the "state" attribute of the <presence> element in the presence document to value "full". PA MUST also include a "version" attribute and it MUST be initialized to value zero.

When the PA generates subsequent notifications, PA SHOULD include only those tuples that have changed compared to the previous notification. It is up to the PA local policy to determine what is considered as a change to the previous state. When the presence document does not include a full presence state, the PA MUST set the "state" attribute value of the <presence> element to "partial".

The PA MUST ensure that all tuple ids used in the presence document are unique. This applies to all tuple ids in <tuple> elements and in <t_id> elements under the <removed> element.

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

- The PA MUST construct the presence information according to the Presence event package [4], with the exception that only the changed tuples are included in the presence document. The PA MUST also add all newly created tuples, if any, to the presence information.

- The PA MUST include a "version" and "state" attributes in the presence document. The PA MUST increment the version number by one compared to the earlier delivered presence document to the watcher associated with a certain subscription.

- When there are changes (e.g. in the authorization) which lead to removal of tuples from the previously delivered presence information the PA MUST construct a partial notification document that lists the tuple ids of the removed tuples in the "removed" element.

- The PA MUST include all the presence information outside the <tuple> elements in each notification, i.e., all the notifications which convey partial presence documents MUST always have that data.

In case the PA receives a subscription refresh or termination request from existing subscription (with any value in the Expires header field) and the PA decides to include a body into the following NOTIFY request the PA MUST build a NOTIFY request that contains a full presence document (i.e. the attribute "state" is set to "full" and the attribute "version" is set to value 0).

4.5 Watcher processing of NOTIFY requests

If the PA decided to use the partial notifications, then the watcher will receive a NOTIFY request with the Content-Type header field value set to ‘application/pidf-partial+xml’.

The watcher will receive the full presence document in the first notification. In this case, the "state" attribute of the <presence> element in the presence document is set to the value "full". When the watcher receives the full presence document it MUST perform the following actions:
The watcher MUST discard all previously received presence information from that particular presentity in the context of current subscription.

The watcher MUST initialize an internal version counter, related to the particular presentity or subscription, to the value of "version" attribute of the <presence> element included in the presence document.

The watcher MUST store the values of all tuple ids together with the content received in the notification. This constitutes the watcher’s local copy of the full presence document.

If the presence document contains a <removed> element the watcher MUST ignore the content of that element.

When the watcher receives subsequent notifications and the notifier has not changed the used content type, and the "state" attribute of the <presence> element includes the value "partial", the watcher MUST construct the presence information according to the following logic:

The watcher MUST compare the "version" attribute of the <presence> element with the version information of a previously received presence document. If the version number is incremented by one, the watcher MUST continue handling the content present in the notification.

The watcher MUST compare the tuple ids with the tuple ids received in previous notifications. If a tuple id in the current notification matches an existing tuple id, the existing tuple is replaced with the newly received tuple. If the tuple id does not match any tuple received in the earlier notifications, the watcher MUST store it as a new tuple.

The watcher MUST remove from the local storage those tuples whose tuple ids are listed in the "removed" element of the presence document.

Tuples whose tuple ids are not listed in the presence document included in the NOTIFY remain unchanged.

If the watcher receives a presence document whose "version" attribute value higher by more than one 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 receives a presence document with the "state" attribute set to "partial" and the "version" attribute value is equal or lower than
the one received in the presence document of a previous notification, it is considered a PA failure and the watcher SHOULD discard the document without further processing.

All the information located outside the <tuple> element must be processed as specified in the SIP presence event package [4] i.e., the watcher MUST replace the existing data with data received in the latest notification.

If the PA changes the content type used in notifications within the existing subscription the watcher MUST discard all the previously received presence information from that particular presentity and process the new content as specified for that content type.

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 the for the partial notification (‘application/pidf-partial+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 represented in the ‘application/pidf-partial+xml’ content type. The following notifications only include the delta of the presence information from the previous NOTIFY request.

<table>
<thead>
<tr>
<th>watcher</th>
<th>presence agent</th>
<th>pua</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 subscribe</td>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>200 OK</td>
<td>&lt;------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>notify</td>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>200 OK</td>
<td>&lt;------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update presence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>notify</td>
<td>200 OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>notify</td>
<td>200 OK</td>
<td></td>
</tr>
</tbody>
</table>
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@somewhere.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-partial+xml;q=1
Contact: sip:user@watcherhost.example.com
Expires: 600

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@somewhere.com>;tag=xfg9
     Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Event: presence
Expires: 600
Contact: sip:server.example.com

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 a first NOTIFY
request that includes a partial notification document that
contains the full state.

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@somewhere.com>;tag=xfg9
From: <sip:resource@example.com>;tag=ffd2
Call-ID: 2010@watcherhost.example.com
Event: presence
Subscription-State: active; expires=599
Max-Forwards: 70
CSeq: 8775 NOTIFY
Contact: sip:server.example.com
Content-Type: application/pidf-partial+xml
Content-Length: [replace with real content length]

<?xml version="1.0" encoding="UTF-8"?>
<pidf-part:presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:pidf-part="urn:ietf:params:xml:ns:pidf-partial"
entity="pres:someone@example.com" version="0"
state="full">
<tuple id="sg89ae">
  <status><basic>open</basic>
  </status>
  <contact priority="0.8">tel:09012345678</contact>
</tuple>
<tuple id="cg231jcr">
  <status><basic>open</basic>
  </status>
  <contact priority="1.0">im:pep@example.com</contact>
</tuple>
<tuple id="r1230d">
  <status><basic>closed</basic>
  </status>
  <contact priority="0.9">sip:pep@example.com</contact>
</tuple>
</pidf-part:presence>

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@somewhere.com>;tag=xfg9
From: <sip:resource@example.com>;tag=ffd2
Call-ID: 2010@watcherhost.example.com
CSeq: 8775 NOTIFY

At a later time, the presentity’s presence information
changes: the tuple id "cg231jcr" has changed its
status, a new tuple id "wsqw798jcr" is added and the
tuple id "r1230d" is deleted from the presence
document. The PA generates a NOTIFY request
that includes a partial presence document that includes
the deleted tuple

F5 NOTIFY example.com server -> watcher

NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/TCP server.example.com;
    branch=z9hG4bKna998s1
To: <sip:watcher@somewhere.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=543
Max-Forwards: 70
Contact: sip:server.example.com
Content-Type: application/pidf-partial+xml
Content-Length: [replace with real content length]

<?xml version="1.0" encoding="UTF-8"?>
<pidf-part:presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:pidf-part="urn:ietf:params:xml:ns:pidf-partial"
    entity="pres:someone@example.com" version="1"
    state="partial">
    <tuple id="cg231jcr">
        <status><basic>closed</basic>
        </status>
        <contact priority="1.0">
            im:pep@example.com</contact>
        <note xml:lang="en">This is an update of existing
tuple sent in previous notification</note>
    </tuple>

    <tuple id="wsqw798jcr">
        <status><basic>open</basic>
        </status>
        <contact priority="0.4">
            im:mac@hut.com</contact>
        <note xml:lang="en">This is a completely new
tuple not sent in previous notification</note>
    </tuple>

    <pidf-part:removed><pidf-part:t_id>r1230d</pidf-part:t_id>
6. Security Considerations

This specification relies on presence event package for the SIP [4] and it does not introduce any new protocol functionality. Partial notifications can reveal information what has changed compared to last time when notification was sent. This can make it easier for eavesdropper to know what kind of changes are happening in presentity’s presence information. However, same information can be found if presence event package is used with baseline PIDF [6]. Thus, this specification does not introduce any new security considerations compared to presence event package for the SIP [4].

Presence related security considerations are extensively discussed in the presence event package for the SIP [4] and all those identified security consideration apply to this document as well. Issues described in the presence event package for the SIP [4] are briefly reviewed below.

6.1 Confidentiality

Confidentiality considerations identified in the presence event package for the SIP [4] apply here without any changes.

6.2 Message Integrity and Authenticity

Message Integrity and Authenticity identified in the presence event package for the SIP [4] apply here without any changes.

6.3 Outbound Authentication

Outbound Authentication considerations identified in the presence
event package for the SIP [4] apply here without any changes.

6.4 Replay Prevention

Replay Prevention considerations identified in the presence event package for the SIP [4] apply here without any changes.

6.5 Denial of Service Attacks Against Third Parties

Denial of Service Attacks Against Third Parties considerations identified in the presence event package for the SIP [4] apply here without any changes.

6.6 Denial Of Service Attacks Against Servers

Denial Of Service Attacks Against Servers considerations identified in the presence event package for the SIP [4] apply here without any changes.

7. Acknowledgements

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

8. References

8.1 Normative references


8.2 Informative references


Authors’ Addresses

Mikko Lonnfors
Nokia
Itamerenkatu 11-13 00180
Helsinki
Finland
Phone: +358 71 8008000
EMail: mikko.lonnfors@nokia.com

Jose Costa-Requena
Nokia
Valimotie 9 00380
Helsinki
Finland
Phone: +358 71 8008000
EMail: jose.costa-requena@nokia.com
Eva Leppanen  
Nokia  
P.O BOX 785  
Tampere  
Finland  
Phone: +358 7180 77066  
EMail: eva-maria.leppanen@nokia.com  

Hisham Khartabil  
Nokia  
P.O. Box 321  
Helsinki  
Finland  
Phone: +358 7180 76161  
EMail: hisham.khartabil@nokia.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the IETF’s procedures with respect to rights in IETF Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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