INTERET-DRAFT
draft-lonnofors-simple-partial-notify-00.txt
Expires: July 2003

J.Costa-Requena
Eva Leppanen
Hisham Khartabi
Mikko Lonnfors
Nokia
January 2003

Partial Notification of Presence Information

Status of this Memo

Abstract

A Presence service implemented using SIMPLE has some constraints for delivering presence information to devices with low data processing capabilities, small display, and limited battery power. Other limitations can be caused by the interface between the terminal and the network, i.e. over radio links with high latency and low bandwidth. This memo presents a solution that aids in reducing the impact of those constrains and increasing efficiency, by introducing a new MIME-type æpartial notificationsÆ and a Require header extension æpartial-notificationÆ.

Table of Contents

1 Introduction

SIP extensions for presence [1] allow users (öwatchersö) to subscribe to other users (öpresentitiesö) presence information. The presence information is composed of multiple pieces of data that is delivered to the watcher. The size of the presence information can be large (i.e. an arbitrary number of elements called 'Presence tuples' that convey data). It may not be reasonable to send the complete presence information over low bandwidth and high latency links when only part of that information changes. This may end up in degrading the presence service and causing bad perception at the watcher side. Thus, it is necessary to provide solutions to overcome this problem for the sake of success of the presence service.

Presence based applications in wireless terminals have certain limitations because it is envisioned that the presence service may demand high bandwidth. It is foreseen that the presence information may have a considerable size, especially if large content is included

in presence information. Requirements of wireless environments can also be found in [2].

There are some mechanisms which might be used to help the problem, such as signaling compression [3] and content indirection [4] and

[5]. However, none of the existing solutions are optimal because they set additional requirements on basic network functionalities such as charging and 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 (HTTP) and therefore increases possible security concerns.

This draft presents another approach as a solution to the problem, namely ôPartial Notificationsö. The idea is already identified by the SIP Extensions for Presence document [1] as a potential solution. In general, the partial notification approach means that the Presence Server (PS) delivers to the watchers only that part of the presence information that has changed compared to the previous notification. (See also [2] for requirements). Note that the ôPartial Notificationsö are not IMPP compliant.

This document introduces a new MIME-Type 'application/pidf-partial+xml' and a Require-header extension 'partial-notification'.

1.1 Existing mechanisms

The SIMPLE Working Group has already proposed a set of mechanisms in order to accommodate excessive message size when messages are delivered over links with limited bandwidth.

- SigComp[3]: Signalling compression.
This mechanism provides a reasonable message size reduction because of the text-based nature of SIP. However, this approach has certain limitations when the messages contain binary content and it does not reduce the size after the compression process. This approach also requires having additional functionality in the terminal and in the server. (I.e. the SigComp dictionaries require additional memory that is a scarce resource in small devices, especially when the dictionary includes binary content).

- Content Indirection [4],[5]:
Consists of sending a URL pointing to the content. This approach solves the problem of excessive message size sent in the presence notifications. The notification only includes a URL pointing to the location where the content is stored. The watcher receiving the URL has the option to use amore appropriate data protocol such as HTTP to fetch the content every time a notification is received.

The inconvenience with this mechanism is that the URL has an expiration time and after expiry, the URL becomes obsolete. There are also security concerns since the URL should also include some

security credentials for authentication purposes when fetching the content. In addition to this, the terminal has to fetch the content after receiving the notification, causing delays in receiving the presence information. In some systems this may also complicate the charging system.

- Embedded URL in Presence Content:
Include a URL, that points to the binary content, in the notification. This approach is similar to the previous Content Indirection approach but instead of sending the URL for the whole presence document, the URL included in the presence information is pointing only to some pieces of information (e.g. images or other binary content).

This mechanism would be the most suitable one because if it were used in conjunction with the compression mechanism, it would lead into a reduced size of the message to be delivered in the notification. The inconveniences with this approach are the security, charging and performance problems identified for the 'Content Indirection' approach. Another inconvenience is that he watcher receives the notification and thereafter has to fetch the content that is included in the presence document with URLs.

2 Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [6].

3 Introduction of the partial notification solution

This chapter briefly introduces the current functionality of the Presence service, and gives an overview of the partial notification solution and new items needed to implement it.

The motivation for introducing a partial notification mechanism, in addition to the existing solutions described in Section 1.1, is to mitigate the excessive message size in certain circumstances (notifications with large un-compressed content). The approach is Presence Server (PS) centric in a sense that the watcher, also referred to as subscriber in this document, initiates the negotiation in order to inform the Presence server (PS) about its ability to receive partial notifications. The PS is the ultimate element that decides whether the partial notification capability is used or not in the particular subscription.

3.1 Normal presence server operation

The presence service normally operates so that the watcher sends the SIP SUBSCRIBE method targeted to the presentity. The request is routed up to the PS responsible for storing the presence information of the presentity. The SUBSCRIBE request MAY include an Accept-header field for indicating the supported content types [7].

The PS receives the SUBSCRIBE request and if there is no Accept header indicating supported content types, PS will generate the presence notification using the default PIDF format [8]. PIDF may contain one or multiple 'tuples' and presence document level information. The 'tuples' include a set of elements defined in the presence model [9] for representing the presence information. The presence information is sent to the watcher in the body of the NOTIFY request according to [10]. By default, the presence information contains the full state corresponding to the presence status of the presentity, as allowed by the PS local policy.

3.2 Operation of the partial notification

The proposed mechanism for implementing the partial notification consists of defining a new content type as extension to the existing

'PIDF' format [8]. The new content type is named as 'application/pdif-partial+xml' and it adds new elements in addition to the existing 'PIDF' schema in order to enable the partial notifications. The new elements are 'version' and 'state'.

The 'version' information is a sequence number that is progressively incremented for each watcher when notification is sent. The 'state' indicates the nature of the presence information included in the notified document, whether it contains 'full' or 'partial' state corresponding to the cases when the full presence information or only changed parts of the presence information are delivered. The use of these attributes is similar to watcher information template package [11].

In addition the 'content-type' header of the NOTIFY request also refers to the format type included in the document (i.e. 'CPIM-PIDF+xml' or 'PIDF-partial+xml').

4 Client and server operations

This document assumes that unless otherwise specified in this document normal subscriber and notifier behavior is applied as defined in [12]. The watcher has the same behavior as a subscriber.

4.1.1 Watcher generating SUBSCRIBE requests

The SUBSCRIBE request can be used to negotiate the prefered content type to be used in the notifications. The 'Accept' header is used for this purpose as specified in [7]. When sending a SUBSCRIBE request, the watcher has the following options:

- Watcher may omit the æAcceptÆ header or place default content type as a value of æAcceptÆ header. In this case the watcher is willing to receive only the default presence format.

- Watcher may add the æAcceptÆ header with values: application/cpim-pidf+xml and application/pidf-partial+xml. In this case the watcher is able to process both the default content type and the partial notification content type.

Therefore, the watcher can indicate the content types that it is willing to receive in the NOTIFY using the 'Accept' header. At least the default content type specified for the Presence service MUST be indicated. If the æAcceptÆ header is omitted the default content type (PDIF[8]) MUST be used by the Presence Server.

The watcher MAY include a æRequireÆ header field with the value æpartial-notificationÆ if the watcher wants to be sure that the PS supports and uses the extensions defined in this specification. If PS does understand the extension indicated in the æRequireÆ header field the PS SHOULD reject the subscription or alternatively accept the subscription and send back an error response.

The æSupportedÆ header achieves the same result as the æAcceptÆ header field. Since the æAcceptÆ header MUST be present in a SUSBSCRIBE request that requires a NOTIFY request with a content type that is not the default one, the field presence of æSupportedÆ header is considered to be redundant, but it is not strictly disallowed.

4.1.2 Notifier processing of SUBSCRIBE requests

The Presence Server receives the subscriptions from the watchers and generates the notifications according to [1]. The watcher might have indicated the supported formats of the presence document by listing the content types in the 'Accept' header. The Presence Server MUST compare the content types included in the 'Accept' header with supported ones, and decide which one to use according to its local policy. There are the following cases:

- There is no æAcceptÆ header field present in the request or the æAcceptÆ header field contains the default content type. In this case, the subscription processing proceeds as defined in [1].

- The æAcceptÆ header field contains both æapplication/cpim-pidf+xmlÆ and æapplication/pidf-partial+xmlÆ. Choosing either of these content types is up to the local configuration as defined in [1]

- The æAcceptÆ header field contains both æapplication/cpim-pidf+xmlÆ and æapplication/pidf-partial+xmlÆ the æRequireÆ header field includes æpartial-notificationÆ. If the PS does not support the

æpartial-notificationÆ extension, then it rejects the request as defined in [1]. If the PS supports the extension, it MUST use the æapplication/pidf-partial+xmlÆ content type in all subsequent NOTIFY requests. The semantics for the æRequireÆ header with value æpartial-notificationÆ apply to the content type included in the æAcceptÆ header with value æapplication/pidf-partial+xmlÆ.

Note: If Accept-header field does not include æapplication/pidf-partial+xmlÆ but a Require-header field with æpartial-notificationÆ was present, the PS rejects the subscription.

4.1.3 NOTIFY body for partial notifications

The watchers supporting the partial notification extension described in this document MUST support the 'application/pidf-partial+xml' content-type.

4.1.4 Notifier generation of partial notifications

If the negotiation between the watcher and the PS resulted in the agreement to use partial notifications, then the PS MUST use 'application/pdif-partial+xml' content type in NOTIFY requests.

The PS MUST deliver the full state of the presence information according to [1] in the first notification. In this case, the 'state' element of the presence document MUST be set to the value 'full'. The æversionÆ attribute MUST also be present and it MUST be initialized to value zero.

When the PS generates subsequent notifications, the presence document includes only the tuples that have changed compared to the previous notification. Except in the case when the tuples are removed, the complete presence document is sent. It is up to the local configuration to determine what is considered as change to previous state. The 'state' element's value MUST be set to 'partialÆ.

The PS constructs the presence document according to the following logic:

The delivered presence information is constructed according to [1] in such a way that only changed tuples are delivered. There might also be new tuples added to the presence information because presentity has published more information or because authorization policy has been changed.

In addition, the version number and state element are also included in the presence document. The version number is incremented by one compared to the earlier delivered presence document to the watcher associated with a certain subscription. The version number should follow the COUNTER32 format that after reaching the maximum value it starts from zero [13].

When there are changes (e.g. in the authorisation) which lead to the removal of a tuple from the previously delivered presence information the PS sends the full presence information by setting the value 'full' for the 'state' element, and restarting the version numbering again. The decision to send æfullÆ state is up to the local policy.

4.1.5 Watcher processing of partial notifications

If the negotiation between the watcher and the PS resulted in the agreement to use partial notifications, then the watcher receives 'application/pdif-partial+xml' content type in the subsequent NOTIFY requests.

The watcher receives the full state of the presence information according to [1] in the first notification. In this case, the 'state' element of the presence document has the value 'full'. When the watcher receives full state notifications it MUST perform the following actions:

- Watcher MUST discard all previously received presence information from that particular presentity.

- Watcher MUST initialize internal version counter, related that particular presentity or subscription, to the value received in the notification

- Watcher MUST store the values of all tuple IDs together with the content received in the notification.

When the watcher receives subsequent notifications, the presence document includes only those tuples that have changed compared to the

previous notification. The 'state' element includes the value of 'partial' telling to the watcher that the notification includes partial information. The watcher MUST construct the presence information according to the following logic:

- The version number of the presence document SHOULD be compared with the version number of the previously received presence document. If the version number is incremented by one, the watcher SHOULD continue handling the content present in the notification.

- The Watcher MUST compare tuple IDs to tuple IDs received in previous notifications. If a tuple ID in the notification matches an existing tuple ID, the existing tuple must be replaced with the newly received in the notification. If the tuple ID does not match to the ones received in earlier notifications, it will be stored as new tuple.

- The missing tuples means that they remained unchanged.

In case the watcher receives a partial notification, which has a æversionÆ number that is higher than the stored value by more than one, the watcher assumes that one or more NOTIFY were lost and SHOULD refresh the subscription within the existing dialog in order to receive a complete update (full state) of the presence information. If the watcher receives a notification with æsateÆ value equal æpartialÆ and the æversionÆ number equal or smaller than the previous notification, it is considered a PS failure and the watcher SHOULD refresh the subscription.

OPEN ISSUE: Handling of <note> element. PIDF and application/pidf-partial+xml can have <note> elements located as child elements to <presence> element. It must be specified how these elements are handled when partial notifications are send to watchers. The simplest approach is to send them with the complete information even if they did not change.

5 IANA considerations

This memo calls for IANA to register a new XML namespace URN as defined in [14].

A new content type æapplication/pidf-partial+xmlÆ is defined to represent an XML MIME for the partial presence information content. This specification follows the guidelines of RFC3023 [15].

It may also be needed an IANA registration for the value æpartial-notificationÆ of the æRequireÆ header as a SIP extension and the required information for this registration, as specified in RFC3261 [7], is:

Name: partial-notification

URI:
urn:ietf:params:xml:ns:pidf-partial

Description: This option tag is used in the æRequireÆ header field by a UAC to ensure that partial notifications are honored at the PS.

Registrant Contact: IETF, SIMPLE working group, <simple@ietf.org>

```
BEGIN
   <?xml version="1.0"?>
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
   "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml
   <head>
        <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1"/>
        <title>PIDF extension for partial notifications</title>
   </head>
   <body>
       <h1>Namespace for PIDF extension for partial
notifications</h1>
       <h2>application/pidf-partial+xml</h2>
       <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
    </body>
    </html>
   END
```

6 Examples

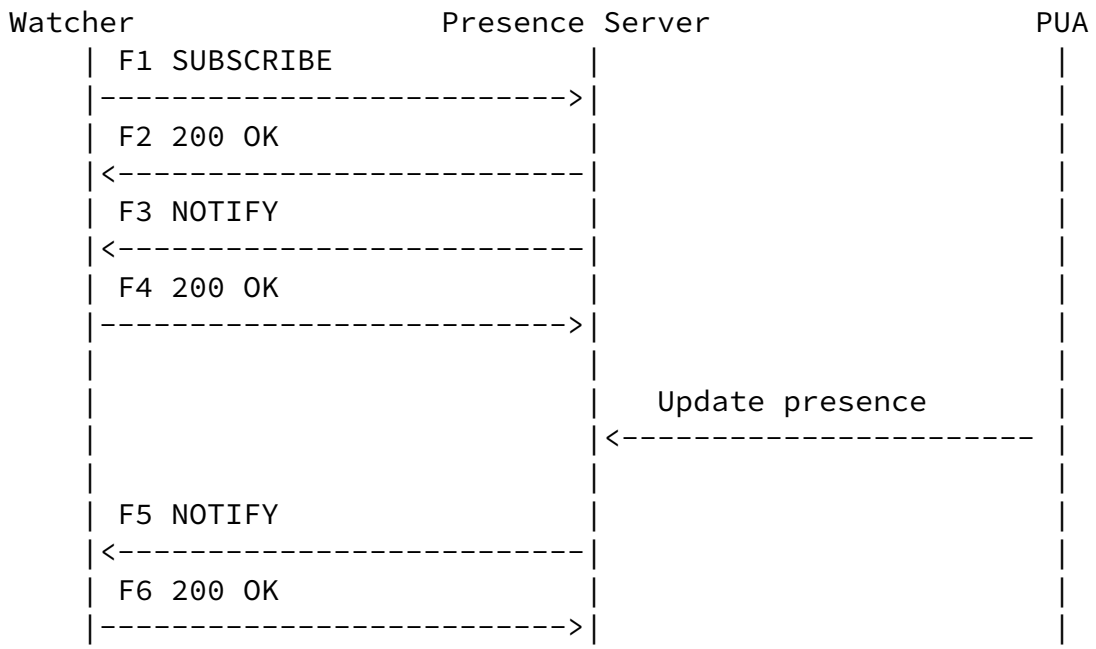
The following flows show examples when applying the proposed partial

notification extension. The document of the æpidf-partial+xmlÆ format mentioned in the message details is constructed according to the XML schema described in the chapter 6.2.

6.1.1 Subscription with partial notification as optional

The watcher sends a SUBSCRIBE request including the default presence format (PIDF) and the PDIF partial extension using the æAcceptÆ

header field. The PS accepts the subscription and based on a local policy it selects to send partial notifications in NOTIFY requests according to the logic described in this document. The first NOTIFY request includes the full state of presence information represented in the æapplication/pidf-partial+xmlÆ contenttypeÆ. The following notifications only include the delta of the presence information from the previous NOTIFY request.



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/cpim-pidf+xml, application/pidf-partial+xml
Contact: user@watcherhost.example.com
Expires: 600
Content-Length: 0
```

F2 200 OK example.com server->watcher

The Presence Server accepts the subscription and based on the local policy it applies the partial notitifcation. (See æContent-TypeÆ= æapplication/pidf-partial+xmlÆ).


```
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
Content-Length: 0
```

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: ..

PIDF-PARTIAL+XML Document with æFULL STATEÆ information:
<?xml version="1.0" encoding="UTF-8"?>
<impp:presence xmlns:impp="urn:ietf:params:xml:ns:pidf-partial"
 entity="pres:someone@example.com" version=ö1ö state=öfullö>

  <impp:tuple id="sg89ae">
    <impp:status><impp:basic>open</impp:basic></impp:status>
    <impp:contact priority="0.8">tel:09012345678</impp:contact>
  </impp:tuple>

  <impp:tuple id="cg231jcr">
    <impp:status><impp:basic>open</impp:basic></impp:status>
    <impp:contact priority="1.0">
          im:pep@nokia.com</impp:contact>
  </impp:tuple>
</impp: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
Content-Length: 0
```

F5 NOTIFY example.com server -> watcher

It is the local policy issue to construct the æPIDF-partial+xmlÆ formated document including the delta from the previous NOTIFY.

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/TCP server.example.com;branch=z9hG4bKna998sl
To: sip:watcher@somewhere.com;tag=xfg9
From: sip:resource@example.com;tag=ffd2
Call-ID: 2010@watcherhost.example.com
```

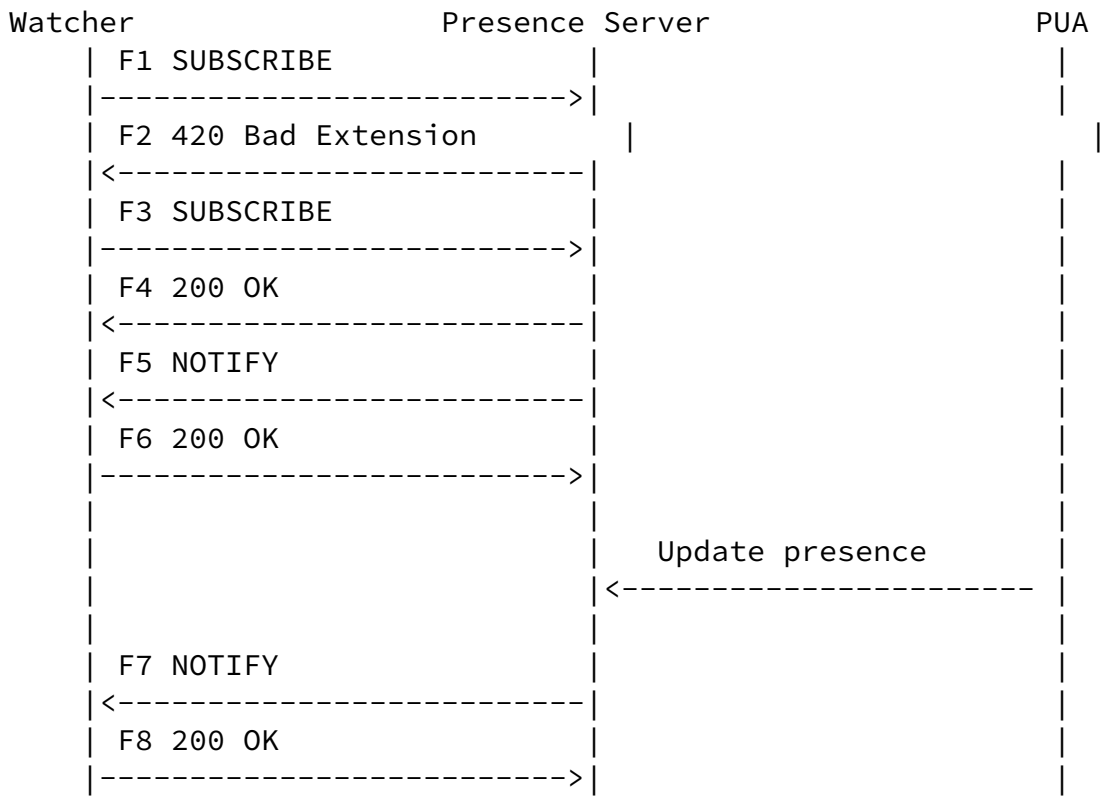
```
CSeq: 8776 NOTIFY
Event: presence
Subscription-State: active;expires=543
Max-Forwards: 70
Contact: sip:server.example.com
Content-Type: application/pidf-partial+xml
Content-Length: ...

New PIDF-PARTIAL+XML Document with æPARTIAL STATEÆ information:
<?xml version="1.0" encoding="UTF-8"?>
<impp:presence xmlns:impp="urn:ietf:params:xml:ns:pidf-partial"
 entity="pres:someone@example.com" version=ö2ö state=öpartialö>

  <impp:tuple id="cg231jcr">
    <impp:status><impp:basic>closed</impp:basic></impp:status>
    <impp:contact priority="1.0">
          im:pep@nokia.com</impp:contact>
    <impp:notexml:lang="en">This is an update of existing tuple
          sent in previous notification</note>
  </impp:tuple>

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


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@somewhere.com;tag=xfg9
From: sip:resource@example.com;tag=ffd2
Call-ID: 2010@watcherhost.example.com
CSeq: 8776 NOTIFY
Content-Length: 0
```

6.1.2 Subscription with partial notification as requirement

The watcher sends SUBSCRIBE request including æpartial-notificationÆ extension in the æRequireÆ header field. The PS does not support the æpartial-notificationÆ extension, so it rejects the subscription. The watcher re-subscribed without including the ærequireÆ header field.



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
Require: partial-notification
Contact: sip:user@watcherhost.example.com
Expires: 600

Content-Length: 0
```

F2 420 Bad Extension example.com server->watcher

The Presence Server does not support the extension indicated in the æRequireÆ header and the subscription is rejected.

```
SIP/2.0 420 OK
Via: SIP/2.0/TCP watcherhost.example.com;branch=z9hG4bKnashds7
  ;received=192.0.2.1
To: sip:resource@example.com
From: sip:watcher@somewhere.com;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Event: presence
Expires: 600
Unsupported: partial-notification
Contact: sip:server.example.com
Content-Length: 0
```

F3 SUBSCRIBE watcher -> example.com server

The watcher makes subscribtion without any requirements for partial notification. The default presence service and PDIF format [1] are applied.

```
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: 17767 SUBSCRIBE
Max-Forwards: 70
Event: presence
```

```
   Contact: sip:user@watcherhost.example.com
   Expires: 600
   Content-Length: 0


F4 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
   From: sip:watcher@somewhere.com;tag=xfg9
   Call-ID: 2010@watcherhost.example.com
   CSeq: 17767 SUBSCRIBE
   Event: presence
   Expires: 600
   Contact: sip:server.example.com
   Content-Length: 0



F5 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/cpim-pidf+xml
   Content-Length: ..

   CPIM-PIDF+XML formatted document:
   <?xml version="1.0" encoding="UTF-8"?>
   <impp:presence xmlns:impp="urn:ietf:params:xml:ns:cpim-pidf"
    entity="pres:someone@example.com">

     <impp:tuple id="sg89ae">
       <impp:status><impp:basic>open</impp:basic></impp:status>
       <impp:contact priority="0.8">tel:09012345678</impp:contact>
     </impp:tuple>

     <impp:tuple id="cg231jcr">
       <impp:status><impp:basic>open</impp:basic></impp:status>
       <impp:contact priority="1.0">
```

```
                im:pep@nokia.com</impp:contact>
          </impp:tuple>

        </impp:presence>
```

F6 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
        Content-Length: 0
```

F7 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@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/cpim-pidf+xml
        Content-Length: ...

        New CPIM-PIDF+XML formatted_document:
        <?xml version="1.0" encoding="UTF-8"?>
        <impp:presence xmlns:impp="urn:ietf:params:xml:ns:cpim-pidf"
         entity="pres:someone@example.com">

          <impp:tuple id="sg89ae">
            <impp:status><impp:basic>open</impp:basic></impp:status>
            <impp:contact priority="0.8">tel:09012345678</impp:contact>
```

```
            </impp:tuple>

            <impp:tuple id="cg231jcr">
              <impp:status><impp:basic>closed</impp:basic></impp:status>
              <impp:contact priority="1.0">
                    im:pep@nokia.com</impp:contact>
```



```
              <impp:note xml:lang="en">This is an update of existing
                      tuple sent in previous notification</note>
            </impp:tuple>

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

          </impp:presence>



       F8 200 OK
          SIP/2.0 200 OK
          Via: SIP/2.0/TCP server.example.com;branch=z9hG4bKna998sl
           ;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: 8776 NOTIFY
          Content-Length: 0
```

7 XML Schema

The XML schema for the æpidf-partial+xmlÆ data format.

```
<?xml version="1.0" encoding="UTF-8"?>
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:pidf-partial"
        xmlns:tns="urn:ietf:params:xml:ns:pidf-partial"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">

    <!-- This import brings in the XML language attribute xml:lang-->
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="http://www.w3.org/2001/xml.xsd"/>

    <xs:element name="presence" type="tns:presence"/>

    <xs:complexType name="presence">
      <xs:sequence>
        <xs:element name="tuple" type="tns:tuple"
           maxOccurs="unbounded"/>
        <xs:element name="note" type="tns:note" minOccurs="0"
```



```
           maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
           maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="entity" type="xs:anyURI" use="required"/>
      <xs:attribute name="version" type="xs:nonNegativeInteger"
                    use="required"/>
      <xs:attribute name="state" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="full"/>
             <xs:enumeration value="partial"/>
            </xs:restriction>
          </xs:simpleType>
      </xs:attribute>
    </xs:complexType>

    <xs:complexType name="tuple">
      <xs:sequence>
        <xs:element name="status" type="tns:status"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="contact" type="tns:contact" minOccurs="0"/>
        <xs:element name="note" type="tns:note" minOccurs="0"
           maxOccurs="unbounded"/>
        <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
```

```
</xs:complexType>

<xs:complexType name="status">
  <xs:sequence>
    <xs:element name="basic" type="tns:basic" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
       maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
 <xs:simpleType name="basic">
  <xs:restriction base="xs:string">
    <xs:enumeration value="open"/>
    <xs:enumeration value="closed"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="contact">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="priority" type="tns:qvalue"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```



```
<xs:complexType name="note">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute ref="xml:lang"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="qvalue">
  <xs:restriction base="xs:decimal">
    <xs:pattern value="0(.[0-9]{0,3})?"/>
    <xs:pattern value="1(.0{0,3})?"/>
  </xs:restriction>
</xs:simpleType>

<!-- Global Attributes -->
<xs:attribute name="mustUnderstand" type="xs:boolean" default="0">
  <xs:annotation>
```

```
        <xs:documentation>
        This attribute may be used on any element within an optional
        PIDF extension to indicate that the corresponding element must
        be understood by the PIDF processor if the enclosing optional
        element is to be handled.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:schema>
```

8 Security Considerations

The security considerations for the presence are considered in [2], and RFC 2779 [5] also outlines them. Furthermore, the PDIF [8] document includes additional security requirements to be considered in the data format for the partial notifications.

9 Acknowledgements

The authors would like to thank Kriztian Kiss, Juha Kalliokulju and Tim Moran for their valuable comments.

10 References

[1] Rosenberg, J., et al, "SIP Extensions for Presence", draft-ietf-simple-presence-07.txt. Internet Draft, May 2002, Work in progress.

[2] K.Kiss, ôRequirements for Presence Service based on 3GPP specifications and wireless environment characheristicsö draft-kiss-simple-presence-wireless-reqs-01.txt. Internet Draft, October 2002. Work in progress.

[3] Price, R., et al, öSignaling Compression (SigComp)ö, RFC 3320, Internet Engineering Task Force, January 2003.

[4] Olson, S.,ö A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messagesö, draft-ietf-sip-content-indirect-mech-01. Internet Draft, November 2002, Work in progress.

[5] Khartabil. H.,ö Congestion safety and Content Indirectionö, draft-khartabil-sip-congestionsafe-ci-01.txt, Internet Draft, October

2002, Work in progress.

[6] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[7] J. Rosenberg, et al. ôSIP: Session Initiation Protocolö. RFC 3261, Internet Engineering Task Force, June 2002.

[8] H. Sugano, S. Fujimoto, et al, "CPIM presence information data format," draft-ietf-impp-cpim-pidf-05.txt. Internet Draft, May 2002. Work in progress.

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

[10] Roach, A., "SIP-Specific Event Notification", RFC 3265, Internet Engineering Task Force, June 2002.

[11] Rosenberg, J.,ö A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)ö, draft-ietf-simple-winfo-package-04.txt, Internet Draft, December 2002. Work in progress.

[12] Day, M., Aggarwal, S., Mohr, G., Vincent,K., "Instant Messaging/ Presence Protocol Requirements", RFC 2779, Internet Engineering Task Force, February 2000.

[13] K. McCloghrie et al, ôStructure of Management Information Version 2 (SMIv2)ö, RFC 2578, STD 58, April 1999.

[14] Mealling, M., "The IETF XML Registry", draft-mealling-iana-xmlns-registry-04, June 2002, work in progress.

[15] M. Murata, S. S. Laurent, and D. Kohn, "XML media types," RFC 3023, Internet Engineering Task Force, Jan. 2001.

11 Author's Addresses

Jose Costa-Requena
Nokia
Valimotie 9

00380 HELSINKI
Finland
Tel: +358-71-8008000
E-mail: jose.costa-requena@nokia.com

Mikko Lonnfors
Nokia Research Center
P.O. Box 407
FIN-00045 NOKIA GROUP
Finland
Tel: +358 50 4836402
Email: mikko.lonnfors@nokia.com

Eva Leppanen
Nokia
P.O Box 785
FIN-33101 Tampere
FINLAND
Tel: +358 7180 77066
Email: eva-maria.leppanen@nokia.com

Hisham Khartabil
Nokia
P.O. Box 321
FIN-00045
NOKIA GROUP
FINLAND
Email: hisham.khartabil@nokia.com
Tel: + 358 7180 76161