SIMPLE WG
Internet-Draft
Expires: December 18, 2003

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

June 19, 2003

Partial Notification of Presence Information
draft-lonnfors-simple-partial-notify-02

Status of this Memo

Copyright Notice

Abstract

A Presence service can have 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 to increase efficiency, by introducing a mechanism called partial notification.

Table of Contents

1. Introduction

SIP extensions for presence [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). 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 degrading the presence service and causing bad perception at the watcher side.

Presence based applications in wireless terminals have certain limitations because it is envisioned that the presence service may demand high bandwidth. Requirements for wireless environments can be found in [12].

There are some mechanisms, such as signaling compression [14] and content indirection [11], [10] that might be used to help in this problem. However, none of the existing solutions are optimal because they 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 (HTTP) and therefore increases possible security concerns.

This draft presents a solution to these problems, called Partial Notifications. Requirements for this mechanism are presented in [3]. Other set of requirements can be found from [12]. The idea is already identified by the SIP Extensions for Presence document [4] as a potential solution.

In general, the partial notification approach means that the Presence Server (PS) delivers to the watchers only those parts of the presence information that have changed compared to the presence information sent in the previous notification. Note that partial notification is not IMPP compliant.

This document introduces a new MIME-Type 'application/pidf-partial+xml'.

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.

3. Introduction of the partial notification mechanism

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.

3.1 Normal presence server operation

The presence service normally operates so that the watcher sends the SIP SUBSCRIBE request targeted to the presentity. The request is routed up to the presence server responsible for terminating the request. The SUBSCRIBE request MAY include an Accept header field for indicating the supported content types [5].

The PS receives the SUBSCRIBE request and if there is no Accept header indicating the supported content types, the PS will generate the presence notification using the default PIDF format [6]. The PIDF may contain one or multiple tuples and presence document level information. The tuples include a set of elements defined in the presence model [2] for representing the presence information. The presence information is sent to the watcher in the body of the NOTIFY request according to [7]. By default, the presence information contains the full state corresponding to the presence status of the presentity, as determinated by the PS local policy and authorization rules.

3.2 Operation of the partial notification

The mechanism for implementing the partial notification consists of defining a new content type. The new content type is named as 'application/pidf-partial+xml'. It is similar to PIDF [6] except that it adds new XML items to <presence> element in order to enable the partial notifications. The new XML attributes are "version" and "state". The new XML element is "removed".

The "version" attribute is a sequence number that is progressively incremented for each watcher when notification is sent. The "state" attribute 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 the watcher information template package [9].

In the scope of this document the partial updates apply only to

<tuple> level XML elements and everything what is contained inside these elements, in other words: tuples are considered to be atomic data elements. This means the when an update is send to a tuple it is assumed that the whole tuple is completely replaced by the new one. All the data which is located outside the <tuple> elements must be processed as specified in [4]. Usually this means that all those XML elements (for example the <note> element) must be included in every notification.

4. Client and server operations

This document assumes that unless otherwise specified in this document the normal subscriber and notifier behavior is applied as defined in [4]. The watcher has the same behavior as a subscriber.

4.1 Content-type for partial notifications

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

4.2 Watcher generating SUBSCRIBE requests

The SUBSCRIBE request can be used to negotiate the preferred content type to be used in the notifications. The Accept header is used for this purpose as specified in [5]. When watcher wants to allow PS to send partial notifications it MUST include the Accept header value 'application/pidf-partial+xml' in the SUBSCRIBE request. The qvalue parameter of the Accept header can be used to indicate the most preferred content type to be used.

4.3 Notifier processing of SUBSCRIBE requests

The Presence Server receives the subscriptions from the watchers and generates the notifications according to [4]. If the watcher has indicated the supported content types in the Accept header the Presence Server compares the content types included in the Accept header with the supported ones, and decides which one to use. If the watcher has used the qvalue parameter of the Accept header for the content types the decision should be based on them. Otherwise the decision is made according to the local policy of the server.

4.4 Notifier generating partial notifications

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

The PS MUST deliver the full state of the presence information according to [4] in the first notification. In this case, the "state" attribute of the <presence> element in the presence document MUST be set to the value "full". The "version" 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. It is up to the local policy to determine what is considered as a change to the previous state. The "state" attribute's value MUST be set to "partial".

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

- The delivered presence information is constructed according to [4] in such a way that only the changed tuples are delivered. New tuples are also be added to the presence information, if any.

- The "version" and "state" attributes 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 so that after reaching the maximum value it starts from zero [15].

- When there are changes (e.g. in the authorization) which lead to removal of tuples from the previously delivered presence information the PS lists the IDs of the removed tuples in the "removed" element.

- All the presence information outside the <tuple> elements MUST be included in each notification, i.e., all the notifications which convey partial notifications MUST always have that data.

4.5 Watcher processing of partial notifications

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

The watcher receives the full state of the presence information according to [4] in the first notification using the partial notifications. In this case, the "state" element of the presence document has the value "full". When the watcher receives the full state notification it MUST perform the following actions:

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

- The watcher MUST initialize an internal version counter, related that particular presentity or subscription, to the value received in the notification.

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

When the watcher receives subsequent notifications and the PS has not changed the used content type, 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" attribute of the <presence> element is compared with the version information in the previously received presence document. If the version number is incremented by one, the watcher continues handling the content present in the notification.

- The Watcher compares tuple IDs to the tuple IDs received in the previous notifications. If a tuple ID in the notification matches an existing tuple ID, the existing tuple is replaced with the newly received in the notification. If the tuple ID does not match to those received in the earlier notifications, it is stored as new tuple.

- If the presence document includes the "removed" element the tuples which IDs are listed are removed from the local storage.

- Tuples whose IDs are missing in the NOTIFY remain unchanged.

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

All information received in the notification which is located outside the <tuple> element must be processed as specified in [4]. I.e., the

watcher must replace the existing data with data received in the latest notification.

In case the PS changes the content type used in notifications within the existing dialog the watcher must discard all previously received presence information from that particular presentity and process the new content as specified for that content type.

5. IANA Considerations

This memo calls for IANA to register a new XML namespace URN per [8]. A new content type 'application/pidf-partial+xml' is defined to represent an XML MIME for the partial presence information content. This specification follows the guidelines of RFC3023 [13].

5.1 URN sub-namespace registration for 'urn:ietf:params:xml:ns:pidf-partial'

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

Description:
This is the XML namespace for XML elements defined by [[[RFCXXXX]]] to describe the 'application/pidf-partial+xml' content type for partial notifications.

Registrant Contact:
IETF, SIMPLE working group, <simple@ietf.org>
Mikko Lonnfors, <mikko.lonnfors@nokia.com>

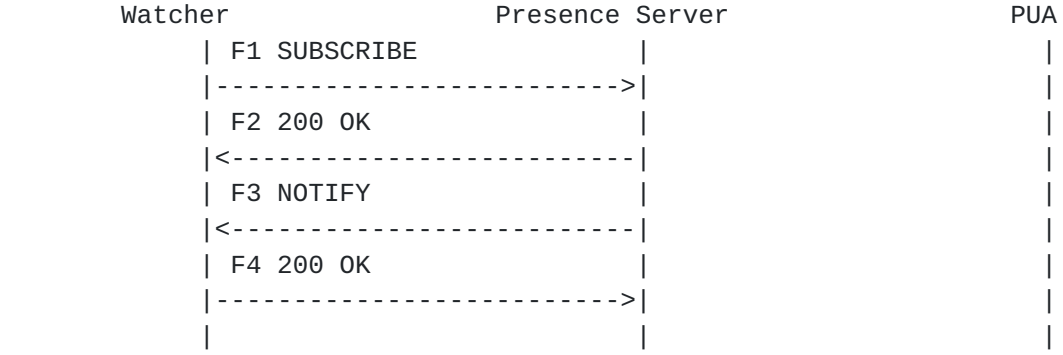
XML:

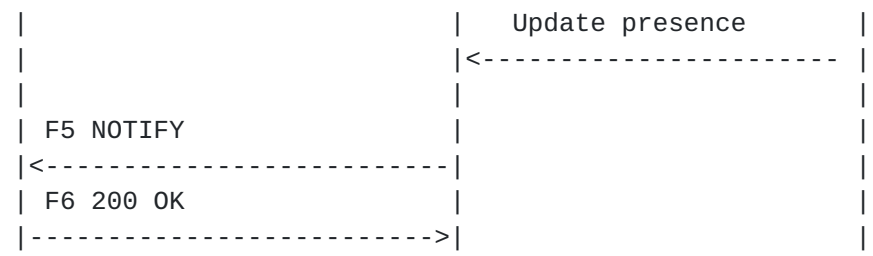
```
BEGIN
    <?xml version="1.0"?>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml
    <head>
         <meta http-equiv="content-type"
         content="text/html;charset=iso-8859-1"/>
         <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 message flow show an example applying the partial notifications mechanism. The document of the 'application/ pidf-partial+xml' format mentioned in the message details is constructed according to the XML schema described in the chapter Section 7.

The watcher sends a SUBSCRIBE request including the default presence format (PIDF) and the content type for the partial notification in the Accept header field. The watcher uses the qvalue parameter to set the preference for receiving partial notifications. The PS accepts the subscription and based on the qvalue information 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.





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;q=0.3, application/pidf-
partial+xml;q=1
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 qvalue information in the Accept header uses the partial notitifcations. (See that the value 'application/pidf-partial+xml' in the Content-Type header).

```
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 FULL STATE 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="1" state="full">

  <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@example.com</impp:contact>
  </impp:tuple>
  <impp:tuple id="r1230d">
    <impp:status><impp:basic>closed</impp:basic></impp:status>
    <impp:contact priority="0.9">
          sip:pep@example.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 'PIDF-partial+xml' formated document including the delta from the previous NOTIFY. Note that the tuple which id was "r1230d" was deleted.

```
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 PARTIAL STATE 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="2" state="partial">

<impp:removed><impp:t_id>r1230d</impp:t_id></impp:removed>

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

7. XML Schema

The XML schema for the 'pidf-partial+xml' 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:element name="removed" type="tns:removed_tuple" minOccurs="0"
            maxOccurs="1"/>
         <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>
```

```
        <xs:complexType name="removed_tuple">
        <xs:sequence>
           <xs:element name="t_id" type="xs:ID" minOccurs="1"
           maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:complexType>

        <!-- 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

Presence information may contain highly sensitive information about the presentities. Presence related security considerations are extensively discussed in [4] and all those identified security consideration apply to this document as well. Partial notification mechanism does not add anything new which (in term of the security) is not already specified in [4] and [5]. Thus no new security considerations are introduced here.

9. Acknowledgements

The authors would like to thank Jyrki Aarnos, Jonathan Rosenberg, Dean Willis, Kriztian Kiss, Juha Kalliokulju and Tim Moran for their valuable comments.

10. Changes

10.1 Changes since -00

- Section comparing the existing mechanisms removed.
- Solution on the removal of the tuple changed. Texts, XML Schema and examples modified accordinly.
- Reduntant texts and issues specified in other specifications

removed.

- Other editorial changes, e.g., more normative style, references devided into Informal and Normative.

10.2 Changes since -01

- Open issues concerning the usage of Require header was solved by the qvalue, and caused changes were incorporated to the text.

- Open issue concerning the change of content type in mid-dialog incorporated to the text.

Normative references

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

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

[3] Lonnfors, M., Leppanen, E., Requena, J. and H. Khartabil, "Requirements for Efficient Delivery of Presence Information", draft-ietf-simple-presinfo-deliv-reg-00 (work in progress), April 2003.

[4] Rosenberg, J., "SIP Extensions for Presence", draft-ietf-simple-presence-10 (work in progress), January 2003.

[5] Rosenberg, J., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[6] Sugano, H., "CPIM presence information data format", draft-ietf-impp-cpim-pidf-08 (work in progress), May 2003.

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

Informative references

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

[9] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", draft-ietf-simple-winfo-package-05 (work in progress), January

2003.

[10] Khartabil, H., "Congestion safety and Content Indirection", draft-khartabil-sip-congestionsafe-ci-02 (work in progress), March 2003.

[11] Olson, S., "Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", draft-ietf-sip-content-indirect-mech-02 (work in progress), February 2003.

[12] Kiss, K., "Requirements for Presence Service based on 3GPP specifications and wireless environment characheristics", draft-kiss-simple-presence-wireless-reqs-01 (work in progress), February 2003.

[13] Murata, M., "XML media types", RFC 3023, January 2001.

[14] Price, R., "Signaling Compression (SigComp)", RFC 3320, January 2003.

[15] McCloghrie, K., "Structure of Management Information Version 2 (SMIv2)", RFC 2578, April 1999.

Authors' Addresses

Mikko Lonnfors
Nokia Research Center
Itamerenkatu 11-13 00180
Helsinki
Finland

Phone: +358 50 4836402
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

Full Copyright Statement

Acknowledgement

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