Internet Engineering Task Force J. Elwell
Internet Draft Siemens

draft-elwell-sipping-redirection-reason-00.txt
Expires: November 2004 May 2004

Indicating redirection reasons in SIP

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3667.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress. "

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/lid-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This examines the need for signalling additional information concerning the reason for redirection in SIP and proposes two possible solutions.

Table of Contents

1 Introduction

Central to SIP [2] is the concept of redirecting or retargeting a request by a proxy, whereby the Request-URI in the original request is replaced before forwarding the request on the next hop. Sometimes this is due to normal rerouting behaviour of the proxy (e.g., resolving an address-of-record URI to a contact URI). At other times it is due to more application-related reasons, e.g., where a user has made arrangements for calls to that user under certain conditions to be forwarded to a different destination. Also retargeting can be performed as a result of a 3xx response from a redirect server. Different 3xx response codes reflect different reasons for rejecting the request.

The History-Info header [3] provides a means for conveying information about a retarget to the final destination UAS and also back to the UAC. In addition to providing the retargeted-from and retargeted-to URIs for each recorded retarget, this header also conveys a reason by means of the Reason header. The Reason header accompanies the retargeted-from URI and reflects the reason why attempts to reach that target failed, normally in the form of the SIP response code concerned.

However, there is nothing in either a 3xx response or the History-Info header to indicate an explicit reason for the redirection request or the retarget respectively. At present the reason is implicit in the reason for failure of the request to the original target. Sometimes this might give an accurate picture of what is happening, but not always. Consider the following cases:

1. A device acts as a redirect server because it is busy. None of the 3xx response codes can reflect that the reason for retargeting to the URI given in the Contact header of the 3xx response is because the existing target is busy.

2. A device acts as a redirect server because it alerts the user but fails to get a reply within a certain time. None of the 3xx response codes can reflect that the reason for retargeting to the URI given in the Contact header of the 3xx response is because the existing target failed to answer.

3. A proxy is scripted to retarget requests without first attempting to forward them to the original target. Retargeting may be unconditional or based on certain conditions such as date, time, the source of the request or caller preferences. Because it does this without forwarding the request to the original target, no SIP response code is applicable.

4. A proxy is scripted to perform hunting or distribution of calls among a number of different targets. When forwarding a request to a target selected from a list of candidate targets, the reason for retargeting is because of hunting or distribution, rather than because of any failure of the existing target.

5. In the hunting or distribution scenario above, forwarding a request to one target from the list of candidate targets may fail for a particular reason (e.g., busy), leading to selection of another target from the list. However, the reason for retargeting is because of hunting or distribution, not specifically because the previous target had a certain condition.

This seems to point to a need to convey in a 3xx response or a History-Info header the reason for selecting the retargeted-to URI. Candidate reasons are:

CFI, "Call forwarding immediate" - immediate retargeting without forwarding the request to the retargeted-from URI;

CFB, "Call forwarding busy" - retargeting because the retargeted-from URI is busy;

CFNR, "Call forwarding no reply" - retargeting because there was no reply at the retargeted-from URI;

CD, "Call deflection" - retargeting because the user at the retargeted-from URI made a request in real time for retargeting;

HUNT, "Hunting" - selection of the target by means of hunting or distribution;

NORMAL "Normal redirection" (default) - normal retargeting of a request.

Note that selection of the new target may depend on several other conditions (e.g., relating to date, time, the source of the request or caller preferences), but the reasons suggested above should be sufficient to convey the main circumstance leading to the retarget.

Two candidate solutions are discussed below.

2 Candidate solutions

2.1 Solution 1 - add a new "protocol" value to the Reason header

New reasons could be achieved by adding a new "protocol" value in the Reason header. For example, assume a session was initiated to sip:+14084953756@foo.com;user=phone.

Assuming the entity sending the INVITE supports the History-Info header, the INVITE would look like this:

```
INVITE  sip:+14084953756@foo.com;user=phone SIP/2.0
From: "Mr. Whatever" <whatever@foo.com>;tag=2
To: <sip:+14084953756@foo.com;user=phone>
Call-ID: 12345600@foo.com
CSeq: 1 INVITE
History-Info: <sip:+14084953756@foo.com;user=phone>;index=1
...
```

The call is then redirected to a contact URI <sip:+44123456789@foo.com;user=phone> in a 302 response. The response would be as follows:

```
SIP/2.0 302 Moved temporarily
From: "Mr. Whatever" <whatever@foo.com>;tag=2
To: <sip:+14084953756@foo.com;user=phone>;tag=3
Call-ID: 12345600@foo.com
CSeq: 1 INVITE
Contact: <sip:+44123456789@foo.com;user=phone>
Reason: Redirection; cause=CFI
?
```

The call would be retargeted to the contact URI. The first History-Info header would be augmented with the two reasons for retargeting (302 and CFI)). A second History-Info header would be added with the new retargeted-to Request-URI:

```
INVITE  sip:+44123456789@foo.com;user=phone SIP/2.0
From: "Mr. Whatever" <whatever@foo.com>;tag=2
To: <sip:+14084953756@foo.com;user=phone>
Call-ID: 12345600@foo.com
CSeq: 1 INVITE
History-Info: <sip:+14084953756@foo.com;user=phone?Reason: SIP;
cause=302; text="Moved temporarily"?Reason: Redirection;
cause=CFI>;index=1, <sip:+44123456789@foo.com;user=phone>;index=2
```

The "index 1" entry indicates that the call to +1-408-495-3756 was retargeted because of SIP response code 302 and redirection reason CFI.

The "index 2" entry indicates that the call to +44-123456789 has not yet been further retargeted.

For the case where the proxy initiates retargeting (not as a result of a 3xx response from a redirect server), the proxy itself would

need to generate the Reason header with Redirection;cause=CFI for inclusion in the index 2 URI in History-Info.

This solution would require either a new standards track RFC or a standard published by another organisation to define the new "protocol" value in the Reason header.

There is an impact on History-Info in that History-Info is required to capture the Redirection reason in a Reason header (since it's not part of the Contact URI in this case). In the current History-Info draft, only the SIP response code is captured in a Reason header.

2.2 Solution 2 - add a new redirection-reason parameter to a contact URI

New reasons could be indicated using a new parameter in a URI.

For example, assume a session was initiated to sip:+14084953756@foo.com;user=phone.

Assuming the entity sending the INVITE supports the History-Info header, the INVITE would look like this:

```
INVITE  sip:+14084953756@foo.com;user=phone SIP/2.0
From: "Mr. Whatever" <whatever@foo.com>;tag=2
To: <sip:+14084953756@foo.com;user=phone>
Call-ID: 12345600@foo.com
CSeq: 1 INVITE
History-Info: <sip:+14084953756@foo.com;user=phone>;index=1
...
```

The call is then redirected to a contact URI <sip:+44123456789@foo.com;user=phone;redirection-reason=CFI> in a 302 response. The response would be as follows:

```
SIP/2.0 302 Moved temporarily
From: "Mr. Whatever" <whatever@foo.com>;tag=2
To: <sip:+14084953756@foo.com;user=phone>;tag=3
Call-ID: 12345600@foo.com
CSeq: 1 INVITE
Contact: <sip:+44123456789@foo.com;user=phone;redirection-
reason=CFI>
?
```

The call would be retargeted to the contact URI. The first History-Info header will be augmented with the Redirection reason (302). A second History-Info header is added with the new retargeted Request-URI:

```
INVITE  sip:+44123456789@foo.com;user=phone;redirection-reason=CFI
SIP/2.0
From: "Mr. Whatever" <whatever@foo.com>;tag=2
To: <sip:+14084953756@foo.com;user=phone>
Call-ID: 12345600@foo.com
CSeq: 1 INVITE
History-Info: <sip:+14084953756@foo.com;user=phone?Reason: SIP;
cause=302; text="Moved temporarily">;index=1,
<sip:+44123456789@foo.com;user=phone;redirection-
reason=CFI>;index=2
```

The "index 1" entry indicates that the call to +1-408-495-3756 was retargeted because of SIP response code 302.

The "index 2" entry indicates that the call to +44-123456789 has not yet been further retargeted, but that it was made as a result of a CFI redirection-reason.

For the case where the proxy initiates retargeting (not as a result of a 3xx response from a redirect server), the proxy itself would need to generate the redirection-reason parameter for inclusion in the index 2 URI in History-Info.

This solution has the advantage that the redirection reason is associated with a particular contact URI and would automatically get copied as part of the contact URI into the Request-URI of the retargeted request. It would be backward compatible with existing implementations of History-Info, since it would automatically be copied with the URI into the History-Info header.

A possible disadvantage is that URI parameters are intended to influence a request constructed from the URI. It might be argued that redirection-reason does not meet this requirement.

Note the difference between this and solution 1, whereby the additional reason is placed in the index 1 URI for solution 1 but in the index 2 URI for solution 2. It is arguable which is the more appropriate. Also solution 1 could be adapted to use the index 2 URI, if considered more appropriate.

The SIP and SIPS URIs are extensible in that new parameters can be added and will be ignored by any implementation that does not understand them. There are plans to create an IANA registry for URI parameters (draft-ietf-sip-uri-parameter-reg-01), and this will require that new parameters be defined in an RFC.

There is no impact on the History-Info draft.

3 Conclusions

The SIP community is asked to express its opinions on the two proposed solutions or suggest other alternatives.

4 Acknowledgements

The author would like to acknowledge considerable assistance from Francois Audet and Mary Barnes in drafting this contribution.

5 Author's Addresses

John Elwell
Siemens Communications
Technology Drive
Beeston
Nottingham, UK, NG9 1LA
email: john.elwell@siemens.com

6 Normative References

[1] H. Schulzrinne, D. Oran, G. Camarillo, "The Reason Header for the Session Initiation Protocol (SIP)", RFC 3326.

[2] J. Rosenberg, H. Schulzrinne, et al., "SIP: Session initiation protocol", RFC 3261.

[3] M. Barnes "An Extension to the Session Initiation Protocol for Request History Information", draft-ietf-sipping-history-info-02 (work in progress)

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.

Acknowledgement

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

AAA Working Group
Internet Draft
draft-hiller-uri-list-index-01.com
Category: Standards Track

Tom Hiller
Lucent Technologies
Adam Roach
Dean Willis
dynamicsoft
July 2004

SIP URI List Index

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 [1].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This draft extends the schema of the resource list specified in draft-ietf-simple-xcap-list-usage-01 by defining an index attribute (membercode). It also defines two MIME types that refer to subsets of a resource list. These MIME types can be used to identify subsets of a resource list for use with SIP requests.

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 [2].

Table of Contents

1. Problem Statement

The SIPPING WG is developing mechanisms to by which a SIP request can target a list of recipient URIs (a URI-list). These URI-lists may be transported either within the associated request (a request-contained list) or stored externally and referenced by the request [URI-LIST] [LIST-CONF]. This specification extends the second case by allowing a request to reference a subset of the URIs contained in a referenced URI-list. This is achieved my extending the schema of the URI-list so that each list element is associated with a locally-unique index value, and extending the referencing syntax to allow a request to carry a list reference and a set of index values indicating elements to be selected from the referenced list.

For wireless, this avoids the need for a mobile to have to send a SIP request multiple times over the air, thereby conserving spectrum and extending battery lifetime, both of which are valuable goals in wireless. The 3G wireless technology cdma2000 has various transport mechanisms, some of which only support rather low data rates. The cdma2000 mechanism "1X-RTT", has an effective payload data rate of 8500bps after various physical overhead are deducted from the absolute physical bit rate. So, for 1X-RTT, a kilobyte long SIP request causes one second of over-the-air transport latency, which is a problem for some services, such as push-to-talk conferencing that have tight latency requirements.

Yet another cdma2000 transport mechanism called "short data burst", has severe message length restrictions; for example based on radio engineering considerations, the PPP frames should be under 100 bytes total including PPP overhead. This mechanism also features low data rates as the 1X-RTT mechanism mentioned above.

For these cdma2000 transport mechanisms, it is highly desirable to minimize the length of SIP requests especially for those destinations that are contacted frequently. Based on this, it is desirable to minimize the number of bytes required to transport the URI-lists in the SIP requests, and therefore to define a highly compact means to convey a URI List in SIP Request bodies. This draft proposes an index to elements of the resource list schema [RF]. The index results in shorter SIP requests for SIP applications that require

transport over lower data rate and/or message length restricted cdma2000 transport mechanisms.

2. General Solution

2.1. Summary

This draft adds an attribute called "membercode" to elements of the resource list schema of [RL] and defines two MIME [MIME-1] types to convey (represent) a URI List [URI-LIST] [LIST-CONF] in the body of a SIP request. The MIME types are based on the identity of the user's resource list along with indices (the membercodes) that have been previously stored in a user's resource list. Both MIME types require that the server hosting the list assign membercodes to all URIs of the user's Resource List entries. The MIME type conveys identity of the resource list and the membercodes associated with the URIs on a URI List. The MIME instance replaces the actual URI elements, thereby saving many bytes and reducing over-the-air transport latency.

The membercode is a non-negative integer that is unique within a given resource list. The maximum value (size) of the membercode should be on the order of the number of lists and list entries of the resource list.

2.2. Membercode Attribute Management

The document draft-ietf-simple-xcap-list-usage-01 [RL] states the requirements on XCAP for a client to manipulate the resource list.

An XCAP client does not include the membercode attribute when it creates or modifies a resource list element, as the membercode is optional. Instead the server creates a unique membercode when the resource is created. The server leaves the membercode unchanged when a list or list element is modified. The addition of an index to the resource list is transparent to XCAP. Having the presence list server assigns membercodes to resource list elements avoids conflicts and/or race conditions that could arise due to multiple users creating or modifying resource list elements.

Users of the resource list may subscribe for updates to receive presence notifications [RL-NOTIFY] that carry the assigned membercodes. Also, users may access the resource list directly via XCAP to learn the membercode attributes created. Otherwise, a means to synchronize the membercodes in user devices must be provided by means outside the scope of this document.

In order for a users learn the values of membercode attributes via presence notifications [RL-NOTIFY] the user has to subscribe for notifications and the resource list's "subscribe" flag MUST be set).

2.3. MIME Types

The first MIME, application/resource-lists-indices, is a list of the membercodes of the elements of a URI list.

The second MIME, application/resource-lists-bitmap, is a bit map of the membercodes of the elements of the URI list. In the latter case, a bit set at location 'x' in the bit map corresponds to a membercode of value '2**x".

MIME bodies may be further compressed with procedures that are part of a general SIGCOMP [SIGCOMP] "program".

3. Definition of Membercode Attribute for Resource List

As explained above, this draft adds an attribute called "membercode" to elements of the resource list schema of [RL]. , The resulting schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema targetNamespace="urn:ietf:params:xml:ns:resource-lists"
  xmlns:xcap="urn:ietf:params:xml:ns:xcap-must-understand"
  xmlns="urn:ietf:params:xml:ns:resource-lists"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="urn:ietf:params:xml:ns:xcap-must-
understand"/>
   <xs:element name="resource-lists">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="xcap:mandatory-ns" minOccurs="0"/>
      <xs:element name="list" type="listType" minOccurs="0"
      maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:complexType name="listType">
    <xs:sequence maxOccurs="unbounded">
     <xs:choice>
      <xs:element name="list" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:complexContent>
         <xs:extension base="listType"/>
        </xs:complexContent>
       </xs:complexType>
      </xs:element>
      <xs:element name="external" type="xs:anyURI" minOccurs="0"
       maxOccurs="unbounded"/>
      <xs:element name="entry" type="entryType" minOccurs="0"
       maxOccurs="unbounded"/>
      <xs:element name="entry-ref" type="xs:anyURI" minOccurs="0"
       maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
       maxOccurs="unbounded"/>
```

```
        </xs:choice>
       </xs:sequence>
       <xs:attribute name="name" type="xs:string" use="optional"/>
       <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
       <xs:attribute name="subscribeable" type="xs:boolean"
                  use="optional"/>
       <xs:anyAttribute namespace="##other"/>
       <xs: attribute name="membercode"
                      type="unique positiveInteger" use="optional" />
      </xs:complexType>
      <xs:complexType name="entryType">
        <xs:sequence>
         <xs:element name="display-name" type="display-nameType"
                  minOccurs="0"/>
         <xs:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="optional"/>
        <xs:attribute name="uri" type="xs:anyURI" use="required"/>
        <xs: attribute name="membercode"
                      type="unique positiveInteger" use="optional" />
      </xs:complexType>
      <xs:simpleType name="display-nameType">
       <xs:restriction base="xs:string"/>
      </xs:simpleType>
   </xs:schema>
```

4. IANA Considerations

The document draft-camarillo-uri-list-00.txt defines a "list" parameter for SIP and SIPS URIs that points to an XCAP resource list. This document defines two MIME types to which the list parameter may point and is consistent with [MIME-2] and [MIME-4].

4.1. Index List

The MIME Content-Type is "application/resource-lists-indices" and is a list of membercodes separated by white space. The presence of an membercode in the list means that the associated URI is to be included on the URI list. The MIME type includes the resource list URI.

The URI and membercode are encoded as is encoded in UTF-8. The membercode attributes, which are numbers, are coded as hex digits. The URI and member codes are separated by a white space. The exact efficiency of the encoding of membercodes is less important because a SIGCOMP program can compress these digits, which are represented as characters, to binary numbers.

The ABNF [ABNF] for this MIME type is as follows.

```
resource-lists-indices = (resource-list-URI SP *(membercode SP))

      resource-list-URI = SIP-URI
                        ; this is an SIP URI to the resource list
                        ; see [SIP]
           membercode = *HEXDIG
                        ; the member code is on the list if the
                        ; associated URI is on the URI list
```

Information per [MIME-4] is as follows:

MIME media type name: application

MIME subtype name: resource-lists-indices

Mandatory parameters: none

Optional parameters: none

Encoding considerations: UTF-8

Security considerations: See the security section of this specification

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: SIP Requests with an EXPLODE method based URI list.

Additional Information:

Magic Number: None

File Extension: tbd

Macintosh file type code: tbd

Personal and email address for further information: Tom Hiller, tomhiller@lucent.com

Intended usage: COMMON

Author/Change controller: The IETF

4.2. Bit Map MIME

The MIME Content-Type is an "application/resource-lists-bitmap", and is a binary string whose individual bit positions correspond to the values of membercodes. A bit set in the bit map means the URI

associated with the membercode whose value matches that bit position is on the URI list. The MIME type includes the resource list URI.

If the bit map has fewer bits than the maximum value of the membercode, then URIs corresponding to "missing" bit positions are not included in the URI list. If the bit map has bit positions that do not correspond to membercodes or more bits than the maximum value possible of the membercode, then the "extra" bits MUST be ignored.

The URI and bitmap are encoded as is encoded in UTF-8. The bit flags of the membercode are coded as four bits to a hex digit. Any bits in hex digit for which membercodes do not exist are set to zero, which occurs if the number of bits in the bit map isn't a multiple of four. The bit map positions correspond to the power of two in the resulting hex number. Therefore, in string of hex digits, the most significant bit of the most significant hex digit represents the highest value membercode of the resource list.

The bit map MIME type's ABNF is as follows:

```
 resource-lists-bitmap = (resource-list-URI  *membercode-hex)
     resource-list-URI = SIP-URI
                       ; this is a SIP URI to the resource list
                       ; see [SIP]

       membercode-hex = HEXDIG
                       ; a bit position M of the membercode-hex N
                       ; is set if a URI on the URI list
                       ; has a membercode of value 2**(4*N+M)
                       ; where N starts at 1 (so the first character
                       ; is M=1) and M is value of 2**M in the hex
                       ; character (so the least bit is 2**0).
```

Information per [MIME-4] is as follows:

MIME media type name: application

MIME subtype name: resource-lists-bitmap

Mandatory parameters: none

Optional parameters: none

Encoding considerations: UTF-8

Security considerations: See the security section of this specification

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: SIP Requests with an EXPLODE method based URI list.

Additional Information:

Magic Number: None

File Extension: tbd

Macintosh file type code: tbd

Personal and email address for further information: Tom Hiller, tomhiller@lucent.com

Intended usage: COMMON

Author/Change controller: The IETF

5. Security Considerations

The index proposed herein is a way to access a user on the resource list, which is used to invite people to calls, etc. However, the security of the index is no more nor less important than any other data already contained on the list, and therefore, this document does not imply additional security concerns or considerations.

6. References

6.1. Normative

[ABNF] Crocker, "Augmented BNF for Syntax Specifications: ABNF", RFC2234, November 1997

[LIST-CONF] Conference Establishment Using Request-Contained Lists in the Session Initiation Protocol (SIP), draft-ietf-sipping-uri-list-conferencing-00.txt, July 7, 2004

[IANA] Narten, Alvestrand, "Guidelines for Writing an IANA C Considerations Section in RFCs", BCP 26, RFC 2434, October 1998

[MIME-1] Freed, Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC2045, November 1996

[MIME-2] Freed, Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" RFC2046, November 1996

[MIME-4] Freed et al, "Multipurpose Internet Mail

Extensions (MIME) Part Four: Registration Procedures", RFC2048, November 1996

[RL] Rosenberg, "draft-ietf-simple-xcap-list-usage-02", October 27, 2003

[RL-NOTIFY] Roach, Rosenberg, Campbell, A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists, draft-ietf-simple-event-list-04, June 13, 2003

[SIGCOMP] Price et al, Signaling Compression (SigComp), RFC3320, January 2003

[SIP] Rosenberg et al, "The Session Initiation Protocol", RFC3261, June 2002

[URI-LIST] Camarillo, Roach, Providing a Session Initiation Protocol (SIP) Application Server with a List of URIs, draft-camarillo-uri-list-02.txt, March 27, 2004

7. Acknowledgements

Chris Bennet of Togabi suggested the use of the MIME type that conveys a list of indices.

8. Authors' Addresses

Questions about this memo can be directed to:

Tom Hiller
Lucent Technologies
1960 Lucent Lane
Naperville, IL 60566
USA
Phone: +1 630-979-7673
E-mail: tomhiller@lucent.com

Dean Willis
dynamicsoft Inc.
5100 Tennyson Parkway
Suite 1200
Plano, TX 75028
US
Phone: +1 972 473 5455
E-mail: dean.willis@softarmor.com
URI: http://www.dynamicsoft.com/

Adam Roach
dynamicsoft
5100 Tennyson Pkwy
Suite 1200
Plano, TX 75024
USA
E-mail: adam@dynamicsoft.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 Secretariat.

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 practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved. This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING

TASK FORCE DISCLAIMS 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.

Session Initiation Proposal Investigation Working Group
Internet-Draft
Expires: January 8, 2005

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson

July 10, 2004

Evaluating Scenarios for Session-specific Policies
draft-hilt-sipping-policy-scenarios-00

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 8, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This draft describes detailed call flows for different use cases of session-specific policies. It compares the two approaches that are currently being discussed for session-specific policies, namely the piggyback model and the separate channel model.

Table of Contents

1. Introduction

The concept of session-specific SIP session policies [3] has been around for some time. However, it has proven that the mechanisms for establishing session-specific policies are non-trivial and most likely require to sacrifice some of the requirements defined in [5].

In this draft, we compare two approaches that have been proposed for session-specific policies: the piggyback model and the separate channel model. We analyze detailed call flows of use cases for both models and discuss advantages and drawbacks of each model.

The main purpose of this draft is to spark the discussion about the two models and to come to a conclusion on which if the models is the most appropriate approach for session-specific policies.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, [1] and indicate requirement levels for compliant implementations.

3. Scenario

All use cases in the subsequent sections are based on the following scenario (see Figure 1). The user agent UA A is registered at proxy P A, which is responsible for domain A. UA B is registered at P B in domain B. Both domains A and B are separate and they are connected through a transit network.

It is assumed that user agent and proxy of each domain have a relationship (e.g. UA A is a customer of provider running domain A). It is also assumed that the entities in different domains do not necessarily have a relationship. This corresponds to a scenario where a customer of one provider is establishing a session with a customer of another provider. As a consequence, entities in one domain can't make any assumptions about the capabilities of entities in the other domain. In particular, it can't be assumed that session policies are supported in the other domain. Additionally, it is assumed that entities in one domain are not willing to disclose network internals such as session policies to the other domain.

Figure 1

4. Use Cases

4.1 NAT Traversal

In this scenario, each domain is connected to the public Internet through a NAT. UA A and UA B have local, non-routable addresses. The proxies P A and P B implement MIDCOM [6] agents an control an associated NAT that connects their domain to the Internet.

Session policies are needed to accomplish the following tasks for NAT traversal:

- Enable proxies to examine the media addresses and ports in the session description created by its associated UA (can either be an offer or an answer). This information is needed to configure NAT rules for incoming media traffic.
- Enable proxies to modify the media addresses and ports in the session description created by its associated UA (offer or answer). The modification is needed to replace the local addresses with globally routable addresses at which the associated UA is reachable from outside.
- Enable a proxy to examine the media addresses and ports in the session description created by the remote UA (offer or answer). This information is needed to configure NAT rules for outgoing media traffic.

4.1.1 Piggyback Model

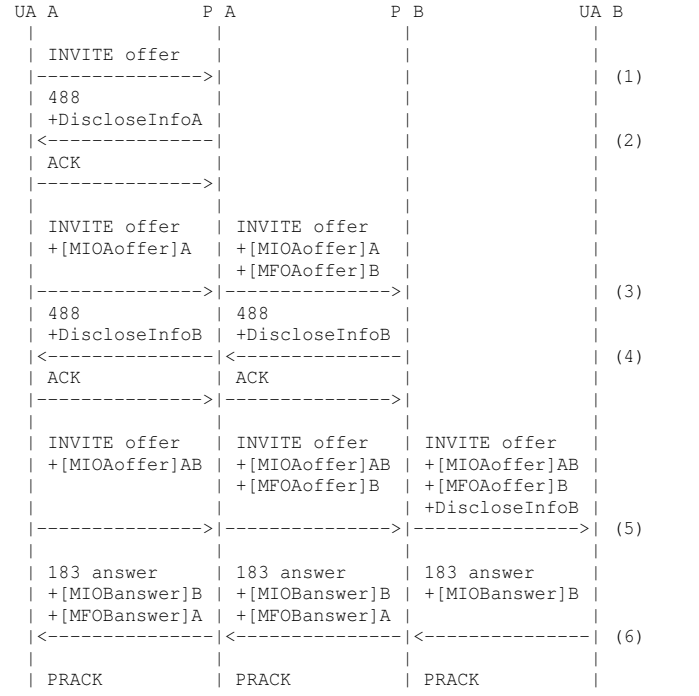
In the piggyback model, session policies are piggybacked on the SIP messages used for the corresponding SDP offer/answer exchange.

4.1.1.1 Offer in Request - Alternative 1

The call flow in Figure 2 describes the piggyback model for INVITE

requests carrying a session description offer. This alternative is based on encryption to protect MIOs and MFOs from being inspected by unauthorized network entities (e.g. in the transit network). It corresponds to the piggyback model that has been discussed so far (e.g. in [3])

It is important to note that this alternative still requires that the UAs on both sides support session-specific policies, even if policies are only used in one domain. In other words, to enable the use of policies between UA A and P A in domain A, UA B in domain B also needs to support policies, even if policies are not used in this domain. Furthermore, encryption can only protect policies from being inspected in the transit network. Entities in both domains must be able to inspect the policies of the other domain.



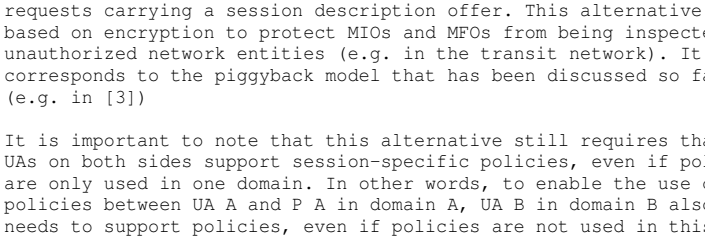


Figure 2

Steps (1) and (2) are needed if P A detects that UA A does not disclose the required aspects of its session description offer in a Media Interface Object A (MIOAoffer). In this case, P A returns a 488 response that requests the disclosure of these aspects. This steps could be avoided, for example, by providing information about what to disclose as part of the device configuration [4].

In step (3) UA A creates Media Interface Object A (MIOAoffer) that discloses the IP addresses and ports it has used in the offer. UA A encrypts MIOAoffer with a key known to P A ([MIOAoffer]A). P A can now perform its MIDCOM functionalities based on the data in MIOAoffer and creates a Media Filter Object for MIOAoffer (MFOAoffer), which contains the external addresses and ports UA B must use to reach UA A. P A encrypts MFOAoffer with a key known to UA B.

In step (4) P B returns a 488 response and asks UA A to disclose the addresses and ports used in the offer. It also asks P A to disclose all policies that affect the addresses and ports in the offer, since these are the addresses and ports that will later be used in the session.

Step (5) is analogous to step (3) except that MIOAoffer and MFOAoffer are now encrypted with a keys known to P B and UA B. Finally, P B asks UA B to disclose the addresses and ports it is going to use in the answer.

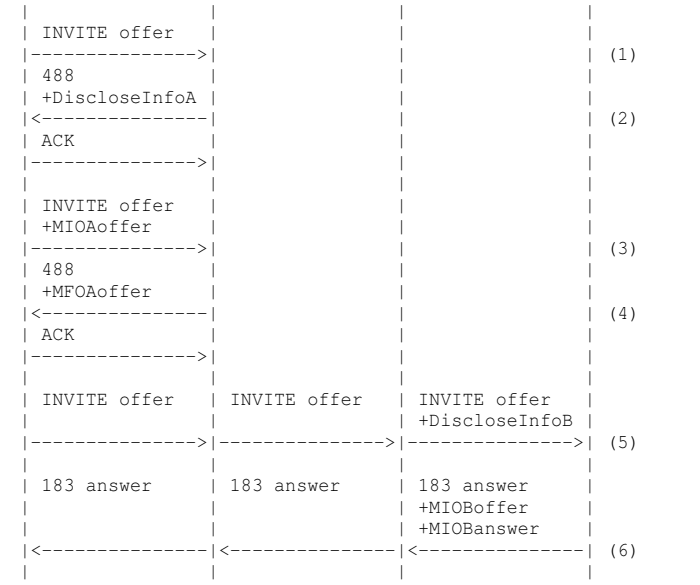
In step (6) UA B has accepted the policies contained in MFOAoffer. It creates a 183 response with its session description answer and a MIOBanswer containing the local IP addresses and ports. UA B encrypts MIOBanswer with a key known to P B. The use of a 183 response instead of a 200 OK later enables UA A to cancel the INVITE transaction if it decides not to accept the requested policies before the INVITE transaction is completed.

P B examines the addresses and ports in MIOBanswer and inserts MFOBanswer containing the external addresses and ports to be used with the session description answer. It encrypts MFOBanswer with a key known to UA A.

In step (7) UA A accepts the policies in MFOBanswer and creates a PRACK. It inserts a MIOAanswer, which contains the addresses and ports it is using to send media to UA B. UA A encrypts MIOAanswer with a key known to P A. Since P A has no policies for the answer, no additional MFOs are needed.

4.1.1.2 Offer in Request - Alternative 2

The call flow in Figure 3 also piggybacks policy information on messages exchanged within a SIP INVITE transaction. In this call flow, these messages are used to exchange policies between UA and proxy. The flow ensures that policy information does not leave the local domain by rejecting messages and removing policy headers.



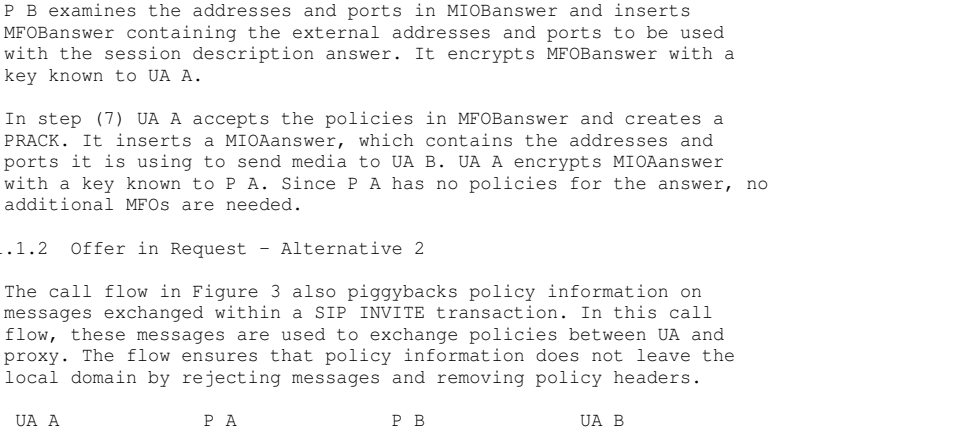


Figure 3

The basic idea of exchanging MIOs and MFOs is the same as in the above flow. Steps (1) - (3) are identical. In step (4) P A returns a MFOAoffer containing the modified addresses and ports for the offer to UA A. UA A can now apply these policies and create a new offer in step (5).

In step (5) P B also requests the disclosure of the addresses used in the offer and answer and receives them from UA B in step (6). Since UA B has not received policies from P B yet, the answer in step (6) is a dummy answer that needs to be updated later.

In step (7) UA A creates a PRACK containing a MIOAanswer which is still based on the dummy answer. P B uses this PRACK message to transmit the addresses and ports it wants UA B to use in its session description to UA B. To make these addresses and ports known to UA A, UA B creates an new offer and sends an UPDATE in step (8) to which UA A responds in step (9). UA A also creates a new MIOAanswer for P A that is now based on the actual session description used in the session.

4.1.1.3 Offer in Response

The piggyback model call flows for INVITEs that carry the session description offer in the response are analogous to the above call

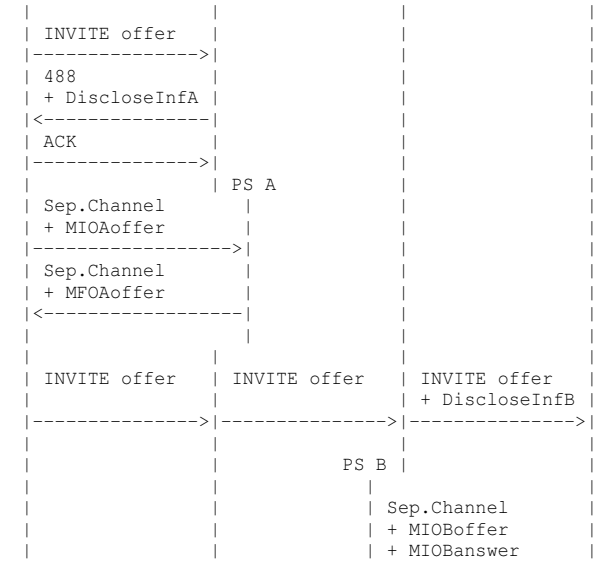
flows. However, these flow are generally more complex that the flows described above for the offer in request scenario.

4.1.2 Separate Channel Model

The idea behind the Separate Channel Model is that user agents retrieve session-specific policies through a separate channel before they create the session description offer/answer. The channel can be implemented in different ways, based on SIP or on another protocol. In this document we simply make the assumption that this channel enables a UA to send a MIO to the policy server and to retrieve a MFO as a response.

4.1.2.1 Offer in Request

The call flow in Figure 4 depicts the separate channel model for INVITE requests carrying a session description offer. PS A and PS B are the policy servers in the respective domains. They can be co-located with the proxies P A and P B but do not have to be.



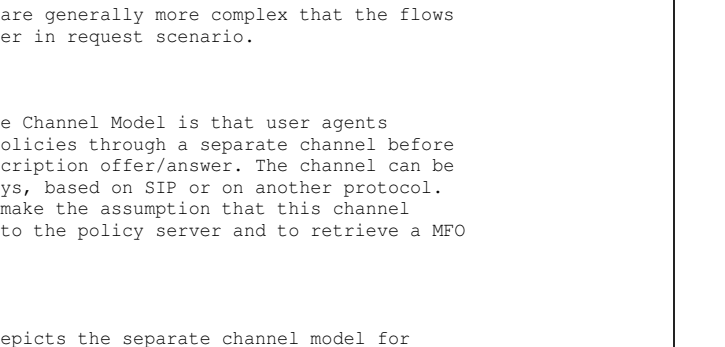


Figure 4

Steps (1) and (2) are needed if P A detects that UA A has not requested policies for the current session before creating the SDP offer. In this case, P A returns a 488 response that contains the address to which UA A should establish a channel to and information about what should be disclosed in an MIO. These steps can be avoided, for example, by providing the information about what to disclosure to where as part of the device configuration.

In step (3) UA A establishes a channel to PS A and submits a MIOAoffer in which it reveals the addresses and ports it is going to use in the offer. PS A uses this information in its function as MIDCOM agent and returns the addresses and ports UA A should include in its offer in an MFOAoffer in step (4).

In step (5) UA A decides to accept the policies in MFOAoffer and creates the offer using the given addresses and ports. P B inserts disclosure information for UA B into this message.

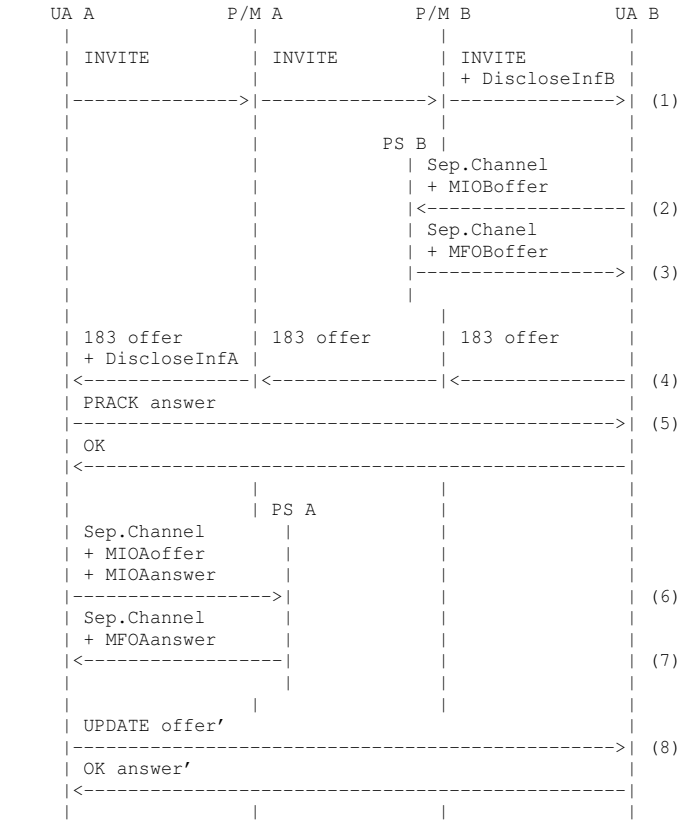
Before creating an answer, UA B retrieves the policies that apply to this session by establishing a channel to its policy server in step (6). It submits the addresses and ports from the offer in MIOBoffer and the addresses and ports it is going to use in its answer in MIOBanswer. PS B returns the addresses and ports to be used in the answer in MFOBanswer in step (7). If UA B decides to accept these policies, it creates an answer in step (8). If not, UA B can return a final response rejecting the INVITE.

In step (9), UA A submits MIOAanswer to the local policy server

disclosing the addresses and ports received in the answer from UA B.

4.1.2.2 Offer in Response

The call flow for an INVITE carrying the offer in the response is depicted in Figure 5. In contrast to call flow Figure 4, UA A has to wait until it receives an offer from UA B before it can retrieve the policies for the current session.



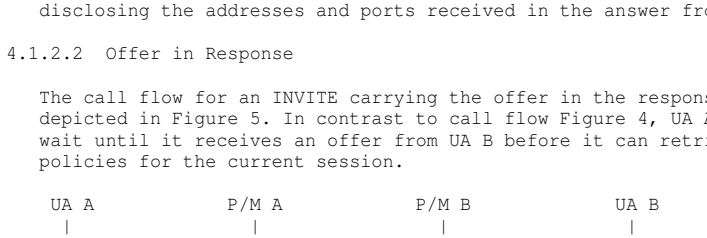


Figure 5

After receiving the 183 response in step (4), UA A must respond immediately with a PRACK to avoid the expiration of timer T1 in UA B and the retransmission of the 183. UA A therefore creates a PRACK with an answer that does not yet consider session-specific policies. It then retrieves the policies for the current session in steps (6) and (7) in which it gets the external addresses and ports from PS A in MFOAanswer. It creates a new offer and sends it to UA B in the UPDATE shown in step (7).

ISSUE: If it can be assumed that UA A and the policy server are located in the same network, there might be enough time for UA A to retrieve policies before generating the PRACK. The sequence of steps would then be (1)-(3),(5)-(6),(4) without a need for the UPDATE in step (7). Is this a reasonable assumption?

4.2 Codec Selection

In this scenario, session-specific policies are used to limit the set of codecs a UA can use. By using session-specific policies, a network provider does not need to reveal the list of allowed codecs to the UA. Instead it can limit the use of certain codecs only if endpoints announce them in an SDP description.

Session policies are needed to accomplish the following tasks for codec selection:

- Enable a proxy to examine the codecs listed in the session description offer (independent of whether the offer was created by the local or the remote UA).
- Enable proxies to remove codecs from the offer (independent of whether the offer was created by the local or the remote UA).

The call flows for both models are analogous to the NAT scenario,

with the difference that the policy servers do not not provide policies for the answer. Instead, they both provide policies for the session description offer. Also, MIOs contain lists of codecs and MFOs identify those codecs that should not be used.

5. Discussion

5.1 Disclosure of Session Descriptions and Policies

In the piggyback model (alternative 1), all MIOs and MFOs travel through the network. End-to-middle and middle-to-end encryption can be used to prevent unauthorized network entities from examining them. However, even with encryption, UAs need to disclose MIOs to all policy-enabled proxies even if they are located in remote networks. Moreover, proxies must disclose their policies to UAs in remote networks and to other proxies that are interested in examining or modifying the same aspect of a session description.

In the piggyback model (alternative 2), the MIOs and MFOs are piggybacked on messaged which are destined at entities outside of the local network. By rejecting messages and removing headers, the proxies keep the MIOs and MFOs within the local network. End-to-middle and middle-to-end encryption can be used to further protect the MIOs and MFOs so that they can't be examined by unauthorized entities even if these packets accidentally leave the local network.

In the separate channel model, UAs exchange MIOs and MFOs on a separate channel directly with the policy server. UAs can therefore disclose different aspects of a session description to each server. Each server can return policies directly to the UA. End-to-end encryption can be used to secure these transmissions. If UA and the policy server are in the same network, the MIOs and MFOs never exit that network.

5.2 UA Support of Policies

In the piggyback model (alternative 1) both UAs need to support policies, even if they are only used in one of the domains.

In the piggyback model (alternative 2) and the separate channel model, it is sufficient if one of the UAs supports policies.

5.3 Re-Use of Document Formats and Mechanisms

The piggyback model (both alternatives) requires that proxy servers insert MFOs into SIP messages. The current standards require the use of headers for this purpose, since a proxy is not allowed to add body

elements to a message. As a consequence, standard document formats that could be used in MIME bodies can't be used for MFOs in the piggyback model. In addition, S/MIME encryption doesn't apply.

In the separate channel model, MIOs and MFOs are exchanged over a separate channel which is potentially able to carry arbitrary documents. This enables the use of existing document formats for MIOs and MFOs and the use of encryption. In particular, the document formats that are defined for session-independent policies [2] can be re-used for session-specific policies. This greatly simplifies UAs which support both types of policies.

5.4 Asynchronous Policies

Some scenarios require that a policy server can update the session policies at any time for ongoing sessions.

In the piggyback model (both alternatives), the exchange of policies is tied to UA initiated offer/answer exchanges of session descriptions (i.e. INVITE, re-INVITE or UPDATE). For this reason, a proxy can't introduce new policies at arbitrary times during a session.

In the separate channel model, the policy server can send updates for the current policy at any time, independent of messages exchanged between the UAs.

5.5 Separation of Tasks

It is generally desirable to develop separate solutions for different tasks. In the piggyback model (both alternatives), the task of exchanging MIOs and MFOs between UA and policy server is coupled to the task of exchanging the offer/answer between UAC and UAS. This increases the complexity of call flows, in particular if the transmission of MIO/MFOs is spread across different SIP transactions, and leads lower re-usability of solutions for each task.

The separate channel model provides a clear separation of tasks.

6 References

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

[2] Hilt, V., Camarillo, G. and J. Rosenberg, "Session-Independent Policies for the Session Initiation Protocol (SIP)", draft-hilt-sipping-session-indep-policy-01 (work in progress), May 2004.

[3] Hilt, V. and J. Rosenberg, "A Framework for Session-Specific Intermediary Session Policies in SIP", draft-hilt-sipping-session-spec-policy-00 (work in progress), September 2003.

[4] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-03 (work in progress), May 2004.

[5] Rosenberg, J., "Requirements for Session Policy for the Session Initiation Protocol (SIP)", draft-ietf-sipping-session-policy-req-01 (work in progress), February 2004.

[6] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A. and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

Authors' Addresses

Volker Hilt
Bell Labs/Lucent Technologies
101 Crawfords Corner Rd
Holmdel, NJ 07733
USA

EMail: volkerh@bell-labs.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Appendix A. Acknowledgements

Many thanks to Jonathan Rosenberg and Allison Mankin.

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.

SIPPING
Internet-Draft
Expires: January 17, 2005

J. Rosenberg
dynamicsoft

July 19, 2004

A Framework for Application Interaction in the Session Initiation Protocol (SIP)

draft-ietf-sipping-app-interaction-framework-02

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 17, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes a framework for the interaction between users and Session Initiation Protocol (SIP) based applications. By interacting with applications, users can guide the way in which they operate. The focus of this framework is stimulus signaling, which allows a user agent to interact with an application without knowledge of the semantics of that application. Stimulus signaling can occur to a user interface running locally with the client, or to a remote user interface, through media streams. Stimulus signaling encompasses a wide range of mechanisms, ranging from clicking on

hyperlinks, to pressing buttons, to traditional Dual Tone Multi Frequency (DTMF) input. In all cases, stimulus signaling is supported through the use of markup languages, which play a key role in this framework.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1] provides the ability for users to initiate, manage, and terminate communications sessions. Frequently, these sessions will involve a SIP application. A SIP application is defined as a program running on a SIP-based element (such as a proxy or user agent) that provides some value-added function to a user or system administrator. Examples of SIP applications include pre-paid calling card calls, conferencing, and presence-based [11] call routing.

In order for most applications to properly function, they need input from the user to guide their operation. As an example, a pre-paid calling card application requires the user to input their calling card number, their PIN code, and the destination number they wish to reach. The process by which a user provides input to an application is called "application interaction".

Application interaction can be either functional or stimulus. Functional interaction requires the user device to understand the semantics of the application, whereas stimulus interaction does not. Stimulus signaling allows for applications to be built without requiring modifications to the user device. Stimulus interaction is the subject of this framework. The framework provides a model for how users interact with applications through user interfaces, and how user interfaces and applications can be distributed throughout a network. This model is then used to describe how applications can instantiate and manage user interfaces.

2. Definitions

SIP Application: A SIP application is defined as a program running on a SIP-based element (such as a proxy or user agent) that provides some value-added function to a user or system administrator. Examples of SIP applications include pre-paid calling card calls, conferencing, and presence-based [11] call routing.

Application Interaction: The process by which a user provides input to an application.

Real-Time Application Interaction: Application interaction that takes place while an application instance is executing. For example, when a user enters their PIN number into a pre-paid calling card application, this is real-time application interaction.

Non-Real Time Application Interaction: Application interaction that takes place asynchronously with the execution of the application. Generally, non-real time application interaction is accomplished through provisioning.

Functional Application Interaction: Application interaction is functional when the user device has an understanding of the semantics of the interaction with the application.

Stimulus Application Interaction: Application interaction is considered to be stimulus when the user device has no understanding of the semantics of the interaction with the application.

User Interface (UI): The user interface provides the user with context in order to make decisions about what they want. The user enters information into the user interface. The user interface interprets the information, and passes it to the application.

User Interface Component: A piece of user interface which operates independently of other pieces of the user interface. For example, a user might have two separate web interfaces to a pre-paid calling card application - one for hanging up and making another call, and another for entering the username and PIN.

User Device: The software or hardware system that the user directly interacts with in order to communicate with the application. An example of a user device is a telephone. Another example is a PC with a web browser.

User Device Proxy: A software or hardware system that a user indirectly interacts through in order to communicate with the application. This indirection can be through a network. An example is a gateway from IP to the Public Switched Telephone Network (PSTN). It acts a user device proxy, acting on behalf of the user on the circuit network.

User Input: The "raw" information passed from a user to a user interface. Examples of user input include a spoken word or a click on a hyperlink.

Client-Local User Interface: A user interface which is co-resident with the user device.

Client-Remote User Interface: A user interface which executes remotely from the user device. In this case, a standardized interface is needed between the user device and the user interface. Typically, this is done through media sessions - audio, video, or application sharing.

Media Interaction: A means of separating a user and a user interface by connecting them with media streams.

Interactive Voice Response (IVR): An IVR is a type of user interface that allows users to speak commands to the application, and hear responses to those commands prompting for more information.

Prompt-and-Collect: The basic primitive of an IVR user interface. The user is presented with a voice option, and the user speaks their choice.

Barge-In: In an IVR user interface, a user is prompted to enter some information. With some prompts, the user may enter the requested information before the prompt completes. In that case, the prompt ceases. The act of entering the information before completion of the prompt is referred to as barge-in.

Focus: A user interface component has focus when user input is provided fed to it, as opposed to any other user interface components. This is not to be confused with the term focus within the SIP conferencing framework, which refers to the center user agent in a conference [13].

Focus Determination: The process by which the user device determines which user interface component will receive the user input.

Focusless User Interface: A user interface which has no ability to perform focus determination. An example of a focusless user interface is a keypad on a telephone.

Presentation Capable UI: A user interface which can prompt the user with input, collect results, and then prompt the user with new information based on those results.

Presentation Free UI: A user interface which cannot prompt the user with information.

Feature Interaction: A class of problems which result when multiple applications or application components are trying to provide services to a user at the same time.

Inter-Application Feature Interaction: Feature interactions that occur between applications.

DTMF: Dual-Tone Multi-Frequency. DTMF refer to a class of tones generated by circuit switched telephony devices when the user presses a key on the keypad. As a result, DTMF and keypad input are often used synonymously, when in fact one of them (DTMF) is merely a means of conveying the other (the keypad input) to a client-remote user interface (the switch, for example).

Application Instance: A single execution path of a SIP application.
Originating Application: A SIP application which acts as a UAC, making a call on behalf of the user.
Terminating Application: A SIP application which acts as a UAS, answering a call generated by a user. IVR applications are terminating applications.
Intermediary Application: A SIP application which is neither the caller or callee, but rather, a third party involved in a call.

3. A Model for Application Interaction

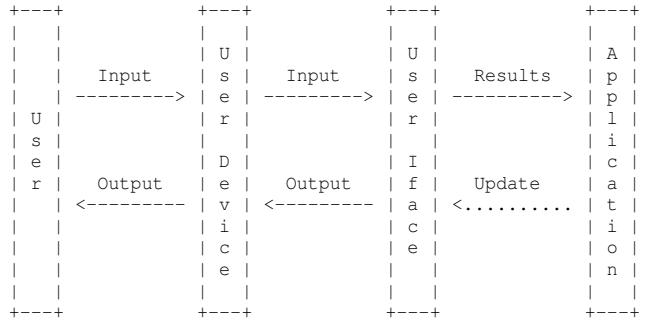


Figure 1: Model for Real-Time Interactions

Figure 1 presents a general model for how users interact with applications. Generally, users interact with a user interface through a user device. A user device can be a telephone, or it can be a PC with a web browser. Its role is to pass the user input from the user, to the user interface. The user interface provides the user with context in order to make decisions about what they want. The user enters information into the user interface. The user interface interprets the information, and passes it as a user interface event to the application. The application may be able to modify the user interface based on this event. Whether or not this is possible depends on the type of user interface.

User interfaces are fundamentally about rendering and interpretation. Rendering refers to the way in which the user is provided context. This can be through hyperlinks, images, sounds, videos, text, and so on. Interpretation refers to the way in which the user interface takes the "raw" data provided by the user, and returns the result to the application as a meaningful event, abstracted from the particulars of the user interface. As an example, consider a pre-paid calling card application. The user interface worries about details such as what prompt the user is provided, whether the voice is male or female, and so on. It is concerned with recognizing the speech that the user provides, in order to obtain the desired information. In this case, the desired information is the calling card number, the PIN code, and the destination number. The application needs that data, and it doesn't matter to the application whether it was collected using a male prompt or a female one.

User interfaces generally have real-time requirements towards the user. That is, when a user interacts with the user interface, the user interface needs to react quickly, and that change needs to be propagated to the user right away. However, the interface between the user interface and the application need not be that fast. Faster is better, but the user interface itself can frequently compensate for long latencies there. In the case of a pre-paid calling card application, when the user is prompted to enter their PIN, the prompt should generally stop immediately once the first digit of the PIN is entered. This is referred to as barge-in. After the user-interface collects the rest of the PIN, it can tell the user to "please wait while processing". The PIN can then be gradually transmitted to the application. In this example, the user interface has compensated for a slow UI to application interface by asking the user to wait.

The separation between user interface and application is absolutely fundamental to the entire framework provided in this document. Its importance cannot be overstated.

With this basic model, we can begin to taxonomize the types of systems that can be built.

3.1 Functional vs. Stimulus

The first way to taxonomize the system is to consider the interface between the UI and the application. There are two fundamentally different models for this interface. In a functional interface, the user interface has detailed knowledge about the application, and is, in fact, specific to the application. The interface between the two components is through a functional protocol, capable of representing the semantics which can be exposed through the user interface. Because the user interface has knowledge of the application, it can be optimally designed for that application. As a result, functional user interfaces are almost always the most user friendly, the fastest and the most responsive. However, in order to allow interoperability between user devices and applications, the details of the functional protocols need to be specified in standards. This slows down innovation and limits the scope of applications that can be built.

An alternative is a stimulus interface. In a stimulus interface, the user interface is generic; totally ignorant of the details of the application. Indeed, the application may pass instructions to the user interface describing how it should operate. The user interface translates user input into "stimulus" - which are data understood only by the application, and not by the user interface. Because they are generic, and because they require communications with the application in order to change the way in which they render information to the user, stimulus user interfaces are usually slower,

less user friendly, and less responsive than a functional counterpart. However, they allow for substantial innovation in applications, since no standardization activity is needed to build a new application, as long as it can interact with the user within the confines of the user interface mechanism. The web is an example of a stimulus user interface to applications.

In SIP systems, functional interfaces are provided by extending the SIP protocol to provide the needed functionality. For example, the SIP caller preferences specification [14] provides a functional interface that allows a user to request applications to route the call to specific types of user agents. Functional interfaces are important, but are not the subject of this framework. The primary goal of this framework is to address the role of stimulus interfaces to SIP applications.

3.2 Real-Time vs. Non-Real Time

Application interaction systems can also be real-time or non-real-time. Non-real interaction allows the user to enter information about application operation asynchronously with its invocation. Frequently, this is done through provisioning systems. As an example, a user can set up the forwarding number for a call-forward on no-answer application using a web page. Real-time interaction requires the user to interact with the application at the time of its invocation.

3.3 Client-Local vs. Client-Remote

Another axis in the taxonomization is whether the user interface is co-resident with the user device (which we refer to as a client-local user interface), or the user interface runs in a host separated from the client (which we refer to as a client-remote user interface). In a client-remote user interface, there exists some kind of protocol between the client device and the UI that allows the client to interact with the user interface over a network.

The most important way to separate the UI and the client device is through media interaction. In media interaction, the interface between the user and the user interface is through media - audio, video, messaging, and so on. This is the classic mode of operation for VoiceXML [4], where the user interface (also referred to as the voice browser) runs on a platform in the network. Users communicate with the voice browser through the telephone network (or using a SIP session). The voice browser interacts with the application using HTTP to convey the information collected from the user.

In the case of a client-local user interface, the user interface runs

co-located with the user device. The interface between them is through the software that interprets the users input and passes them to the user interface. The classic example of this is the web. In the web, the user interface is a web browser, and the interface is defined by the HTML document that it's rendering. The user interacts directly with the user interface running in the browser. The results of that user interface are sent to the application (running on the web server) using HTTP.

It is important to note that whether or not the user interface is local or remote (in the case of media interaction) is not a property of the modality of the interface, but rather a property of the system. As an example, it is possible for a web-based user interface to be provided with a client-remote user interface. In such a scenario, video and application sharing media sessions can be used between the user and the user interface. The user interface, still guided by HTML, now runs "in the network", remote from the client. Similarly, a VoiceXML document can be interpreted locally by a client device, with no media streams at all. Indeed, the VoiceXML document can be rendered using text, rather than media, with no impact on the interface between the user interface and the application.

It is also important to note that systems can be hybrid. In a hybrid user interface, some aspects of it (usually those associated with a particular modality) run locally, and others run remotely.

3.4 Presentation Capable vs. Presentation Free

A user interface can be capable of presenting information to the user (a presentation capable UI), or it can be capable only of collecting user input (a presentation free UI). These are very different types of user interfaces. A presentation capable UI can provide the user with feedback after every input, providing the context for collecting the next input. As a result, presentation capable user interfaces require an update to the information provided to the user after each input. The web is a classic example of this. After every input (i.e., a click), the browser provides the input to the application and fetches the next page to render. In a presentation free user interface, this is not the case. Since the user is not provided with feedback, these user interfaces tend to merely collect information as its entered, and pass it to the application.

Another difference is that a presentation-free user interface cannot support the concept of a focus. As a result, if multiple applications wish to gather input from the user, there is no way for the user to select which application the input is destined for. The input provided to applications through presentation-free user interfaces is more of a broadcast or notification operation, as a

result.

4. Interaction Scenarios on Telephones

In this section, we applied the model of Section 3 to telephones.

In a traditional telephone, the user interface consists of a 12-key keypad, a speaker, and a microphone. Indeed, from here forward, the term "telephone" is used to represent any device that meets, at a minimum, the characteristics described in the previous sentence. Circuit-switched telephony applications are almost universally client-remote user interfaces. In the Public Switched Telephone Network (PSTN), there is usually a circuit interface between the user and the user interface. The user input from the keypad is conveyed used Dual-Tone Multi-Frequency (DTMF), and the microphone input as Pulse Code Modulated (PCM) encoded voice.

In an IP-based system, there is more variability in how the system can be instantiated. Both client-remote and client-local user interfaces to a telephone can be provided.

In this framework, a PSTN gateway can be considered a User Device Proxy. It is a proxy for the user because it can provide, to a user interface on an IP network, input taken from a user on a circuit switched telephone. The gateway may be able to run a client-local user interface, just as an IP telephone might.

4.1 Client Remote

The most obvious instantiation is the "classic" circuit-switched telephony model. In that model, the user interface runs remotely from the client. The interface between the user and the user interface is through media, set up by SIP and carried over the Real Time Transport Protocol (RTP) [16]. The microphone input can be carried using any suitable voice encoding algorithm. The keypad input can be conveyed in one of two ways. The first is to convert the keypad input to DTMF, and then convey that DTMF using a suitance encoding algorithm for it (such as PCMU). An alternative, and generally the preferred approach, is to transmit the keypad input using RFC 2833 [17], which provides an encoding mechanism for carrying keypad input within RTP.

In this classic model, the user interface would run on a server in the IP network. It would perform speech recognition and DTMF recognition to derive the user intent, feed them through the user interface, and provide the result to an application.

4.2 Client Local

An alternative model is for the entire user interface to reside on

the telephone. The user interface can be a VoiceXML browser, running speech recognition on the microphone input, and feeding the keypad input directly into the script. As discussed above, the VoiceXML script could be rendered using text instead of voice, if the telephone had a textual display.

For simpler phones without a display, the user interface can be described by a Keypad Markup Language request document [7]. As the user enters digits in the keypad, they are passed to the user interface, which generates user interface events that can be transported to the application.

4.3 Flip-Flop

A middle-ground approach is to flip back and forth between a client-local and client-remote user interface. Many voice applications are of the type which listen to the media stream and wait for some specific trigger that kicks off a more complex user interaction. The long pound in a pre-paid calling card application is one example. Another example is a conference recording application, where the user can press a key at some point in the call to begin recording. When the key is pressed, the user hears a whisper to inform them that recording has started.

The ideal way to support such an application is to install a client-local user interface component that waits for the trigger to kick off the real interaction. Once the trigger is received, the application connects the user to a client-remote user interface that can play announements, collect more information, and so on.

The benefit of flip-flopping between a client-local and client-remote user interface is cost. The client-local user interface will eliminate the need to send media streams into the network just to wait for the user to press the pound key on the keypad.

The Keypad Markup Language (KPML) was designed to support exactly this kind of need [7]. It models the keypad on a phone, and allows an application to be informed when any sequence of keys have been pressed. However, KPML has no presentation component. Since user interfaces generally require a response to user input, the presentation will need to be done using a client-remote user interface that gets instantiated as a result of the trigger.

It is tempting to use a hybrid model, where a prompt-and-collect application is implemented by using a client-remote user interface that plays the prompts, and a client-local user interface, described by KPML, that collects digits. However, this only complicates the application. Firstly, the keypad input will be sent to both the

media stream and the KPML user interface. This requires the application to sort out which user inputs are duplicates, a process that is very complicated. Secondly, the primary benefit of KPML is to avoid having a media stream towards a user interface. However, there is already a media stream for the prompting, so there is no real savings.

5. Framework Overview

In this framework, we use the term "SIP application" to refer to a broad set of functionality. A SIP application is a program running on a SIP-based element (such as a proxy or user agent) that provides some value-added function to a user or system administrator. SIP applications can execute on behalf of a caller, a called party, or a multitude of users at once.

Each application has a number of instances that are executing at any given time. An instance represents a single execution path for an application. Each instance has a well defined lifecycle. It is established as a result of some event. That event can be a SIP event, such as the reception of a SIP INVITE request, or it can be a non-SIP event, such as a web form post or even a timer. Application instances also have a specific end time. Some instances have a lifetime that is coupled with a SIP transaction or dialog. For example, a proxy application might begin when an INVITE arrives, and terminate when the call is answered. Other applications have a lifetime that spans multiple dialogs or transactions. For example, a conferencing application instance may exist so long as there are any dialogs connected to it. When the last dialog terminates, the application instance terminates. Other applications have a liftime that is completely decoupled from SIP events.

It is fundamental to the framework described here that multiple application instances may interact with a user during a single SIP transaction or dialog. Each instance may be for the same application, or different applications. Each of the applications may be completely independent, in that they may be owned by different providers, and may not be aware of each others existence. Similarly, there may be application instances interacting with the caller, and instances interacting with the callee, both within the same transaction or dialog.

The first step in the interaction with the user is to instantiate one or more user interface components for the application instance. A user interface component is a single piece of the user interface that is defined by a logical flow that is not synchronously coupled with any other component. In other words, each component runs more or less independently.

A user interface component can be instantiated in one of the user agents in a dialog (for a client-local user interface), or within a network element (for a client-remote user interface). If a client-local user interface is to be used, the application needs to determine whether or not the user agent is capable of supporting a client-local user interface, and in what format. In this framework,

all client-local user interface components are described by a markup language. A markup language describes a logical flow of presentation of information to the user, collection of information from the user, and transmission of that information to an application. Examples of markup languages include HTML, WML, VoiceXML, and the Keypad Markup Language (KPML) [7].

Unlike an application instance, which has very flexible lifetimes, a user interface component has a very fixed lifetime. A user interface component is always associated with a dialog. The user interface component can be created at any point after the dialog (or early dialog) is created. However, the user interface component terminates when the dialog terminates. The user interface component can be terminated earlier by the user agent, and possibly by the application, but its lifetime never exceeds that of its associated dialog.

There are two ways to create a client local interface component. For interface components that are presentation capable, the application sends a REFER [6] request to the user agent. The Refer-To header field contains an HTTP URI that points to the markup for the user interface. For interface components that are presentation free (such as those defined by KPML), the application sends a SUBSCRIBE request to the user agent. The body of the SUBSCRIBE request contains a filter, which, in this case, is the markup that defines when information is to be sent to the application in a NOTIFY.

If a user interface component is to be instantiated in the network, there is no need to determine the capabilities of the device on which the user interface is instantiated. Presumably, it is on a device on which the application knows a UI can be created. However, the application does need to connect the user device to the user interface. This will require manipulation of media streams in order to establish that connection.

The interface between the user interface component and the application depends on the type of user interface. For presentation capable user interfaces, such as those described by HTML and VoiceXML, HTTP form POST operations are used. For presentation free user interfaces, a SIP NOTIFY is used. The differing needs and capabilities of these two user interfaces, as described in Section 3.4, is what drives the different choices for the interactions. Since presentation capable user interfaces require an update to the presentation every time user data is entered, they are a good match for HTTP. Since presentation free user interfaces merely transmit user input to the application, a NOTIFY is more appropriate.

Indeed, for presentation free user interfaces, there are two

different modalities of operation. The first is called "one shot". In the one-shot role, the markup waits for a user to enter some information, and when they do, reports this event to the application. The application then does something, and the markup is no longer used. In the other modality, called "monitor", the markup stays permanently resident, and reports information back to an application until termination of the associated dialog.

6. Deployment Topologies

This section presents some of the network topologies in which this framework can be instantiated.

6.1 Third Party Application

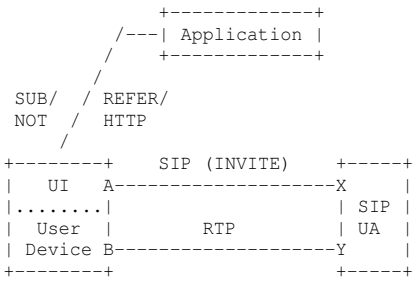


Figure 2: Third Party Topology

In this topology, the application that is interested in interacting with the users exists outside of the SIP dialog between the user agents. In that case, the application learns about the initiation and termination of the dialog, along with the dialog identifiers, through some out of band means. One such possibility is the dialog event package [15]. Dialog information is only revealed to trusted parties, so the application would need to be trusted by one of the users in order to obtain this information.

At any point during the dialog, the application can instantiate user interface components on the user device of the caller or callee. It can do this either using SUBSCRIBE or REFER, depending on the type of user interface (presentation capable or presentation free).

6.2 Co-Resident Application

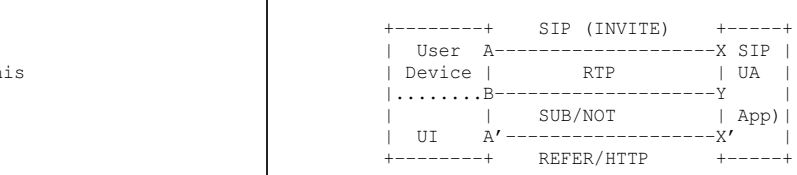


Figure 3: Co-Resident Topology

In this deployment topology, the application is co-resident with one of the user agents (the one on the right in the picture above). This application can install client-local user interface components on the other user agent, which is acting as the user device. These components can be installed using either SUBSCRIBE, for presentation free user interfaces, or REFER, for presentation capable ones.

If the application resides in the called party, it is called a terminating application. If it resides in the calling party, it is called an originating application.

This kind of topology is common in protocol converter and gateway applications.

6.3 Third Party Application and User Device Proxy

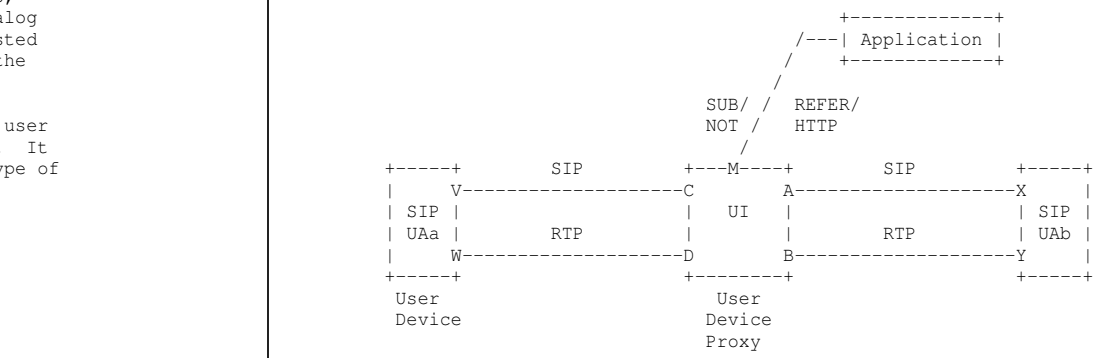


Figure 4: User Device Proxy Topology

In this deployment topology, there is a third party application as in

Section 6.1. However, instead of installing a user interface component on the end user device, the component is installed in an intermediate device, known as a User Device Proxy. From the perspective of the actual user device (on the left), the User Device Proxy is a client remote user interface. As such, media, typically transported using RTP (including RFC 2833 for carrying user input), is sent from the user device to the client remote user interface on the User Device Proxy. As far as the application is concerned, it is installing what it thinks is a client local user interface on the user device, but it happens to be on a user device proxy which looks like the user device to the application.

The user device proxy will need to terminate and re-originate both signaling (SIP) and media traffic towards the actual peer in the conversation. The User Device Proxy is a media relay in the terminology of RFC 3550 [16]. The User Device Proxy will need to monitor the media streams associated with each dialog, in order to convert user input received in the media stream to events reported to the user interface. This can pose a challenge in multi-media systems, where it may be unclear on which media stream the user input is being sent. As discussed in RFC 3264 [18], if a user agent has a single media source and is supporting multiple streams, it is supposed to send that source to all streams. In cases where there are multiple sources, the mapping is a matter of local policy. In the absence of a way to explicitly identify or request which sources map to which streams, the user device proxy will need to do the best job it can. This specification RECOMMENDS that the User Device Proxy monitor the first stream (defined in terms of ordering of media sessions within a session description). As such, user agents SHOULD send their user input on the first stream, absent a policy to direct it otherwise.

6.4 Proxy Application

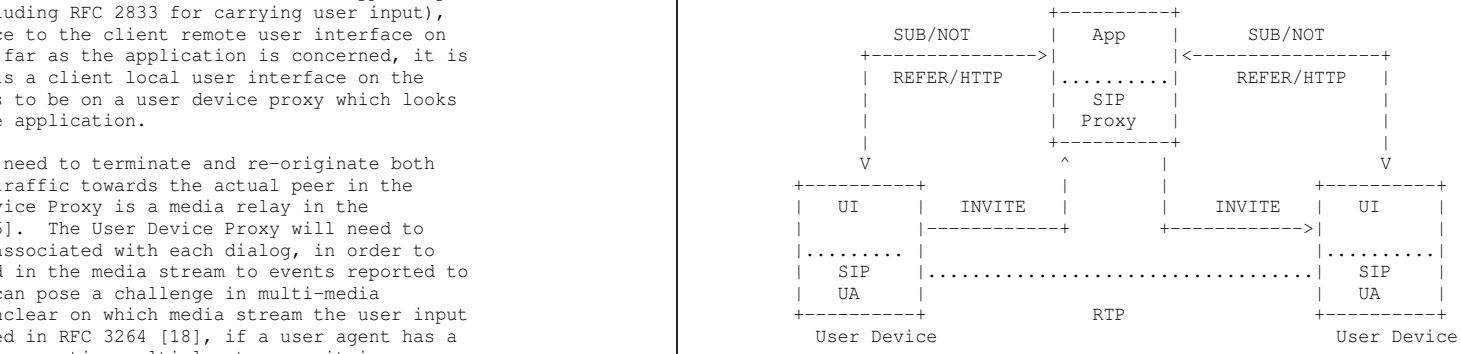


Figure 5: Proxy Application Topology

In this topology, the application is co-resident with a transaction stateful, record-routing proxy server on the call path between two user devices. The application uses SUBSCRIBE or REFER to install user interface components on one or both user devices.

This topology is common in routing applications, such as a web-assisted call routing application.

7. Application Behavior

The behavior of an application within this framework depends on whether it seeks to use a client-local or client-remote user interface.

7.1 Client Local Interfaces

One key component of this framework is support for client local user interfaces.

7.1.1 Discovering Capabilities

A client local user interface can only be instantiated on a user agent if the user agent supports that type of user interface component. Support for client local user interface components is declared by both the UAC and a UAS in its Accept, Allow, Contact and Allow-Event header fields of dialog-initiating requests and responses. If the Allow header field indicates support for the SIP SUBSCRIBE method, and the Allow-Event header field indicates support for the kpml package [7], and the Supported header field indicates that its Contact URI is a GRUU [8], it means that the UA can instantiate presentation free user interface components. In this case, the application MAY push presentation free user interface components according to the rules of Section 7.1.2. The specific markup languages that can be supported are indicated in the Accept header field.

If the Allow header field indicates support for the SIP REFER method, the Supported header field indicates support for the "refer-context" extension described below, and the Contact header field contains UA capabilities [5] that indicate support for the HTTP URI scheme, it means that the UA supports presentation capable user interface components. In this case, the application MAY push presentation capable user interface components to the client according to the rules of Section 7.1.2. The specific markups that are supported are indicated in the Accept header field.

A third party application that is not present on the call path will not be privy to these headers in the dialog requests that pass by. As such, it will need to obtain this capability information in other ways. One way is through the registration event package [19], which can contain user agent capability information provided in REGISTER requests [5].

7.1.2 Pushing an Initial Interface Component

Generally, we anticipate that interface components will need to be

created at various different points in a SIP session. Clearly, they will need to be pushed during session setup, or after the session is established. A user interface component is always associated with a specific dialog, however.

An application MUST NOT attempt to push a user interface component to a user agent until it has determined that the user agent has the neccesary capabilities and a dialog has been created. In the case of a UAC, this means that an application MUST NOT push a user interface component for an INVITE initiated dialog until the application has seen a request confirming the receipt of a dialog-creating response. This could be an ACK for a 200 OK, or a PRACK for a provisional response [2]. For SUBSCRIBE initiated dialogs, it MUST NOT push a user interface component until the application has seen a 200 OK to the NOTIFY request. For a user interface component on a UAS, the application MUST NOT push a user interface component for an INVITE initiated dialog until it has seen a dialog-creating response from the UAS. For a SUBSCRIBE initiated dialog, it MUST NOT push a user interface component until it has seen a NOTIFY request from the notifier.

To create a presentation capable UI component on the UA, the application sends a REFER request to the UA. This REFER MUST be sent to the Globally Routable UA URI (GRUU) [8] advertised by that UA in the Contact header field of the dialog initiating request or response sent by that UA. Note that this REFER request creates a separate dialog between the application and the UA. The Refer-To header field of the REFER request MUST contain an HTTP URI that references the markup document to be fetched.

Furthermore, it is essential for the REFER request to be correlated with the dialog to which the user interface component will be associated. This is necessary for authorization and for terminating the user interface components when the dialog terminates. To provide this context, this specification defines the "context" header field parameter as an extension to the Refer-To heder field. The grammar for this header field parameter is:

```
refer-to-ctxt     = "context" EQUAL DQUOTE local-tag "," remote-tag
                    "," callid DQUOTE    ; callid defined in RFC 3261
                    ;; NOTE: any DQUOTEs inside callid MUST be escaped
                    ;; using quoted pair
local-tag         = token
remote-tag        = token

Refer-To          = ("Refer-To" / "r") HCOLON ( name-addr / addr-spec ) *
     (SEMI (generic-param / refer-to-ctxt))
```

The application MUST include the context header field parameter in the REFER request. The remote-tag MUST be set to the remote tag of the dialog as seen by the user device. The local-tag MUST be set to the local tag of the dialog as seen by the user device. The callid MUST be set to the Call-ID of the dialog as seen by the device. Since the callid grammar allows it to contain double quotes, any such double quotes MUST be represented with a quoted pair.

Since the "context" parameter in the Refer-To header field must be understood by the UA to process the request, this specification defines a new SIP option tag, "refer-context". A REFER request generated by an application MUST include a Require header field with this option tag value. Fortunately, the application will know ahead of time whether this extension is supported, as discussed in Section 7.1.1.

To create a presentation free user interface component, the application sends a SUBSCRIBE request to the UA. The SUBSCRIBE MUST be sent to the GRUU advertised by the UA. This SUBSCRIBE request creates a separate dialog. The SUBSCRIBE request MUST use the KPML [7] event package. The Event header field MUST contain parameters which identify the particular dialog that the interface component is being instantiated against. The body of the SUBSCRIBE request contains the markup document that defines the conditions under which the application wishes to be notified of user input.

In both cases, the REFER or SUBSCRIBE request SHOULD include a display name in the From header field which identifies the name of the application. For example, a prepaid calling card might include a From header field which looks like:

From: "Prepaid Calling Card" <sip:prepaid@example.com>

Any of the SIP identity assertion mechanisms that have been defined, such as [10] and [12] are applicable to these requests as well.

7.1.3 Updating an Interface Component

Once a user interface component has been created on a client, it can be updated. The means for updating it depends on the type of UI component.

Presentation capable UI components are updated using techniques already in place for those markups. In particular, user input will cause an HTTP POST operation to push the user input to the application. The result of the POST operation is a new markup that the UI is supposed to use. This allows the UI to updated in response

to user action. Some markups, such as HTML, provide the ability to force a refresh after a certain period of time, so that the UI can be updated without user input. Those mechanisms can be used here as well. However, there is no support for an asynchronous push of an updated UI component from the appliciation to the user agent. A new REFER request to the same GRUU would create a new UI component rather than updating any components already in place.

For presentation free UI, the story is different. The application MAY update the filter at any time by generating a SUBSCRIBE refresh with the new filter. The UA will immediately begin using this new filter.

7.1.4 Terminating an Interface Component

User interface components have a well defined lifetime. They are created when the component is first pushed to the client. User interface components are always associated with the SIP dialog on which they were pushed. As such, their lifetime is bound by the lifetime of the dialog. When the dialog ends, so does the interface component.

However, there are some cases where the application would like to terminate the user interface component before its natural termination point. For presentation capable user interfaces, this is not possible. For presentation free user interfaces, the application MAY terminate the component by sending a SUBSCRIBE with Expires equal to zero. This terminates the subscription, which removes the UI component.

A client can remove a UI component at any time. For presentation capable UI, this is analagous to the user dismissing the web form window. There is no mechanism provided for reporting this kind of event to the application. The application MUST be prepared to time out, and never receive input from a user. For presentation free user interfaces, the UA can explicitly terminate the subscription. This will result in the generation of a NOTIFY with a Subscription-State header field equal to "terminated".

7.2 Client Remote Interfaces

As an alternative to, or in conjunction with client local user interfaces, an application can make use of client remote user interfaces. These user interfaces can execute co-resident with the application itself (in which case no standardized interfaces between the UI and the application need to be used), or it can run separately. This framework assumes that the user interface runs on a host that has a sufficient trust relationship with the application.

As such, the means for instantiating the user interface is not considered here.

The primary issue is to connect the user device to the remote user interface. Doing so requires the manipulation of media streams between the client and the user interface. Such manipulation can only be done by user agents. There are two types of user agent applications within this framework - originating/terminating applications, and intermediary applications.

7.2.1 Originating and Terminating Applications

Originating and terminating applications are applications which are themselves the originator or the final recipient of a SIP invitation. They are "pure" user agent applications - not back-to-back user agents. The classic example of such an application is an interactive voice response (IVR) application, which is typically a terminating application. It is a terminating application because the user explicitly calls it; i.e., it is the actual called party. An example of an originating application is a wakeup call application, which calls a user at a specified time in order to wake them up.

Because originating and terminating applications are a natural termination point of the dialog, manipulation of the media session by the application is trivial. Traditional SIP techniques for adding and removing media streams, modifying codecs, and changing the address of the recipient of the media streams, can be applied. Similarly, the application can directly authenticate itself to the user through S/MIME, since it is the peer UA in the dialog.

7.2.2 Intermediary Applications

Intermediary applications are, at the same time, more common than originating/terminating applications, and more complex. Intermediary applications are applications that are neither the actual caller or called party. Rather, they represent a "third party" that wishes to interact with the user. The classic example is the ubiquitous pre-paid calling card application.

In order for the intermediary application to add a client remote user interface, it needs to manipulate the media streams of the user agent to terminate on that user interface. This also introduces a fundamental feature interaction issue. Since the intermediary application is not an actual participant in the call, how does the user interact with the intermediary application, and its actual peer in the dialog, at the same time? This is discussed in more detail in Section 9.

8. User Agent Behavior

8.1 Advertising Capabilities

In order to participate in applications that make use of stimulus interfaces, a user agent needs to advertise its interaction capabilities.

If a user agent supports presentation capable user interfaces, it MUST support the REFER method, along with the "context" extension defined here. It MUST include, in all dialog initiating requests and responses, an Allow header field that includes the REFER method and and the Supported header field that includes the value "refer-context". Furthermore, the UA MUST support the SIP user agent capabilities specification [5]. The UA MUST be capable of being REFER'd to an HTTP URI. It MUST include, in the Contact header field of its dialog initiating requests and responses, a "schemes" Contact header field parameter include the http URI scheme. The UA MUST include, in all dialog initiating requests and responses, an Accept header field listing all of those markups supported by the UA. It is RECOMMENDED that all user agents that support presentation capable user interfaces support HTML.

If a user agent supports presentation free user interfaces, it MUST support the SUBSCRIBE [3] method. It MUST support the KPML [7] event package. It MUST include, in all dialog initiating requests and responses, an Allow header field that includes the SUBSCRIBE method. It MUST include, in all dialog initiating requests and responses, an Allow-Events header field that lists the KPML event package. The UA MUST include, in all dialog initiating requests and responses, an Accept header field listing those event filters it supports. At a minimum, a UA MUST support the "application/kpml-request+xml" MIME type.

For either presentation free or presentation capable user interfaces, the user agent MUST support the GRUU [8] specification. The Contact header field in all dialog initiating requests and responses MUST contain a GRUU. The UA MUST include a Supported header field which contains the "gruu" option tag.

Because these headers are examined by proxies which may be executing applications, a UA that wishes to support client local user interfaces should not encrypt them.

8.2 Receiving User Interface Components

Once the UA has created a dialog (in either the early or confirmed states), it MUST be prepared to receive a SUBSCRIBE or REFER request

against its GRUU. If the UA receives such a request prior to the establishment of a dialog, the UA MUST reject the request.

A user agent SHOULD attempt to authenticate the sender of the request. The sender will generally be an application, and therefore the user agent is unlikely to ever have a shared secret with it, making digest authentication useless. However, authenticated identities can be obtained through other means, such as [10].

A user agent MAY have pre-defined authorization policies which permit applications which have authenticated themselves with a particular identity, to push user interface components. If such a set of policies are present, it is checked first. If the application is authorized, processing proceeds.

If the application has authenticated itself, but it is not explicitly authorized or blocked, this specification RECOMMENDS that the application be automatically authorized if it can prove that it was either on the call path, or is trusted by one of the elements on the call path. An application proves this to the user agent by presenting it with the dialog identifiers in the SUBSCRIBE or REFER request. In the case of SUBSCRIBE, those identifiers are present in the Event header field [7]. In the case of REFER, those identifiers are present in the "context" parameter of the Refer-To header field.

Because of the dialog identifiers serve as a tool for authorization, a user agent compliant to this framework SHOULD use dialog identifiers that are cryptographically random, with at least 128 bits of randomness. It is recommended that this randomness be split between the Call-ID and From header field tag in the case of a UAC.

Furthermore, to ensure that only applications resident in or trusted by on-path elements can instantiate a user interface component, a user agent compliant to this specification SHOULD use the sips URI scheme for all dialogs it initiates. This will guarantee secure links between all of the elements on the signaling path.

If the dialog was not established with a sips URI, or the user agent did not choose cryptographically random dialog identifiers, then the application MUST NOT automatically be authorized, even if it presented valid dialog identifiers. A user agent MAY apply any other policies in addition to (but not instead of) the ones specified here in order to authorize the creation of the user interface component. One such mechanism would be to prompt the user, informing them of the identity of the application and the dialog it is associated with. If an authorization policy requires user interaction, the user agent SHOULD respond to the SUBSCRIBE or REFER request with a 202. In the case of SUBSCRIBE, if authorization is not granted, the user agent

SHOULD generate a NOTIFY to terminate the subscription. In the case of REFER, the user agent MUST NOT act upon the URI in the Refer-To header field until user authorization was obtained.

If an application does not present a valid dialog identifier in its REFER or SUBSCRIBE request, the user agent MUST reject the request with a 403 response.

If a REFER request to an HTTP URI was authorized, the UA executes the URI and fetches the content to be rendered to the user. This instantiates a presentation capable user interface component. If a SUBSCRIBE was authorized, a presentation free user interface component was instantiated.

8.3 Mapping User Input to User Interface Components

Once the user interface components are instantiated, the user agent must direct user input to the appropriate component. In the case of presentation capable user interfaces, this process is known as focus selection. It is done by means that are specific to the user interface on the device. In the case of a PC, for example, the window manager would allow the user to select the appropriate user interface component that their input is directed to.

For presentation free user interfaces, the situation is more complicated. In some cases, the device may support a mechanism that allows the user to select a "line", and thus the associated dialog. Any user input on the keypad while this line is selected are fed to the user interface components associated with that dialog.

Otherwise, for client local user interfaces, the user input is assumed to be associated with all user interface components. For client remote user interfaces, the user device converts the user input to media, typically conveyed using RFC 2833, and sends this to the client remote user interface. This user interface then needs to map user input from potentially many media streams into user interface events. The process for doing this is described in Section 6.3.

8.4 Receiving Updates to User Interface Components

For presentation capable user interfaces, updates to the user interface occur in ways specific to that user interface component. In the case of HTML, for example, the document can tell the client to fetch a new document periodically. However, this framework does not provide any additional machinery to asynchronously push a new user interface component to the client.

For presentation free user interfaces, an application can push an update to a component by sending a SUBSCRIBE refresh with a new filter. The user agent will process these according to the rules of the event package.

8.5 Terminating a User Interface Component

Termination of a presentation capable user interface component is a trivial procedure. The user agent merely dismisses the window (or equivalent). The fact that the component is dismissed is not communicated to the application. As such, it is purely a local matter.

In the case of a presentation free user interface, if the user wishes to cease interacting with the application, it SHOULD generate a NOTIFY request with a Subscription-State equal to "terminated" and a reason of "rejected". This tells the application that the component has been removed, and that it should not attempt to re-subscribe.

9. Inter-Application Feature Interaction

The inter-application feature interaction problem is inherent to stimulus signaling. Whenever there are multiple applications, there are multiple user interfaces. When the user provides an input, to which user interface is the input destined? That question is the essence of the inter-application feature interaction problem.

Inter-application feature interaction is not an easy problem to resolve. For now, we consider separately the issues for client-local and client-remote user interface components.

9.1 Client Local UI

When the user interface itself resides locally on the client device, the feature interaction problem is actually much simpler. The end device knows explicitly about each application, and therefore can present the user with each one separately. When the user provides input, the client device can determine to which user interface the input is destined. The user interface to which input is destined is referred to as the application in focus, and the means by which the focused application is selected is called focus determination.

Generally speaking, focus determination is purely a local operation. In the PC universe, focus determination is provided by window managers. Each application does not know about focus, it merely receives the user input that has been targeted to it when its in focus. This basic concept applies to SIP-based applications as well.

Focus determination will frequently be trivial, depending on the user interface type. Consider a user that makes a call from a PC. The call passes through a pre-paid calling card application, and a call recording application. Both of these wish to interact with the user. Both push an HTML-based user interface to the user. On the PC, each user interface would appear as a separate window. The user interacts with the call recording application by selecting its window, and with the pre-paid calling card application by selecting its window. Focus determination is literally provided by the PC window manager. It is clear to which application the user input is targeted.

As another example, consider the same two applications, but on a "smart phone" that has a set of buttons, and next to each button, an LCD display that can provide the user with an option. This user interface can be represented using the Wireless Markup Language (WML).

The phone would allocate some number of buttons to each application. The prepaid calling card would get one button for its "hangup"

command, and the recording application would get one for its "start/stop" command. The user can easily determine which application to interact with by pressing the appropriate button. Pressing a button determines focus and provides user input, both at the same time.

Unfortunately, not all devices will have these advanced displays. A PSTN gateway, or a basic IP telephone, may only have a 12-key keypad. The user interfaces for these devices are provided through the Keypad Markup Language (KPML). Considering once again the feature interaction case above, the pre-paid calling card application and the call recording application would both pass a KPML document to the device. When the user presses a button on the keypad, to which document does the input apply? The user interface does not allow the user to select. A user interface where the user cannot provide focus is called a focusless user interface. This is quite a hard problem to solve. This framework does not make any explicit normative recommendation, but concludes that the best option is to send the input to both user interfaces unless the markup in one interface has indicated that it should be suppressed from others. This is a sensible choice by analogy - its exactly what the existing circuit switched telephone network will do. It is an explicit non-goal to provide a better mechanism for feature interaction resolution than the PSTN on devices which have the same user interface as they do on the PSTN. Devices with better displays, such as PCs or screen phones, can benefit from the capabilities of this framework, allowing the user to determine which application they are interacting with.

Indeed, when a user provides input on a focusless device, the input must be passed to all client local user interfaces, AND all client remote user interfaces, unless the markup tells the UI to suppress the media. In the case of KPML, key events are passed to remote user interfaces by encoding them in RFC 2833 [17]. Of course, since a client cannot determine if a media stream terminates in a remote user interface or not, these key events are passed in all audio media streams unless the KPML request document is used to suppress.

9.2 Client-Remote UI

When the user interfaces run remotely, the determination of focus can be much, much harder. There are many architectures that can be deployed to handle the interaction. None are ideal. However, all are beyond the scope of this specification.

10. Intra Application Feature Interaction

An application can instantiate a multiplicity of user interface components. For example, a single application can instantiate two separate HTML components and one WML component. Furthermore, an application can instantiate both client local and client remote user interfaces.

The feature interaction issues between these components within the same application are less severe. If an application has multiple client user interface components, their interaction is resolved identically to the inter-application case - through focus determination. However, the problems in focusless user interfaces (such as a keypad) generally won't exist, since the application can generate user interfaces which do not overlap in their usage of an input.

The real issue is that the optimal user experience frequently requires some kind of coupling between the differing user interface components. This is a classic problem in multi-modal user interfaces, such as those described by Speech Application Language Tags (SALT). As an example, consider a user interface where a user can either press a labeled button to make a selection, or listen to a prompt, and speak the desired selection. Ideally, when the user presses the button, the prompt should cease immediately, since both of them were targeted at collecting the same information in parallel. Such interactions are best handled by markups which natively support such interactions, such as SALT, and thus require no explicit support from this framework.

11. Example Call Flow

This section shows the operation of a call recording application. This application allows a user to record the media in their call by clicking on a button in a web form. The application uses a presentation capable user interface component that is pushed to the caller.

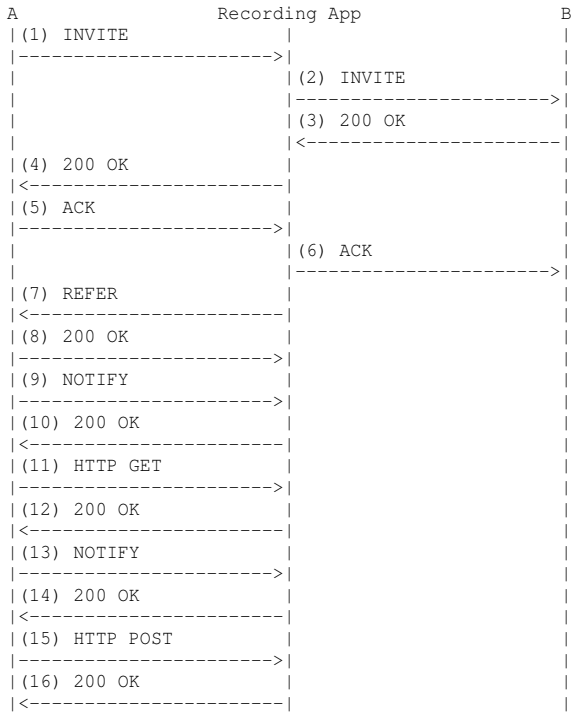


Figure 8

First, the caller, A, sends an INVITE to setup a call (message 1). Since the caller supports the framework, and can handle presentation

capable user interface components, it includes the Supported header field indicating that the GRUU extension and the REFER context extension are understood, Allow indicating that REFER is understood, and a Contact header field that includes the "schemes" header field parameter.

```
INVITE sips:B@example.com SIP/2.0
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.com>
Call-ID: faif9ahhs9dd8==-sd98ajzz@host.example.com
CSeq: 1 INVITE
Max-Forwards: 70
Supported: gruu, refer-context
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Contact: <sips:bad998asd8asd0000a@example.com>;schemes="http,sip,sips"
Content-Length: ...
Content-Type: application/sdp

--SDP not shown--
```

The proxy acts as a recording server, and forwards the INVITE to the called party (message 2):

```
INVITE sips:B@pc.example.com SIP/2.0
Record-Route: <sips:app.example.com;lr>
Via: SIP/2.0/TLS app.example.com;branch=z9hG4bK97sh
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.com>
Call-ID: faif9ahhs9dd8==-sd98ajzz@host.example.com
CSeq: 1 INVITE
Max-Forwards: 69
Supported: gruu, refer-context
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Contact: <sips:bad998asd8asd0000a@example.com>;schemes="http,sip,sips"
Content-Length: ...
Content-Type: application/sdp

--SDP not shown--
```

B accepts the call with a 200 OK (message 3). It does not support the framework, and so the various header fields are not present.

```
SIP/2.0 200 OK
Record-Route: <ssip:app.example.com;lr>
Via: SIP/2.0/TLS app.example.com;branch=z9hG4bK97sh
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.com>;tag=7777
Call-ID: faif9ahhs9dd8==-sd98ajzz@host.example.com
CSeq: 1 INVITE
Contact: <sips:B@pc.example.com>
Content-Length: ...
Content-Type: application/sdp

--SDP not shown--
```

This 200 OK is passed back to the caller (message 4):

```
SIP/2.0 200 OK
Record-Route: <sips:app.example.com;lr>
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.com>;tag=7777
Call-ID: faif9ahhs9dd8==-sd98ajzz@host.example.com
CSeq: 1 INVITE
Contact: <sips:B@pc.example.com>
Content-Length: ...
Content-Type: application/sdp

--SDP not shown--
```

The caller generates an ACK (message 5).

```
ACK sips:B@pc.example.com
Route: <sips:app.example.com;lr>
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz9
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.com>;tag=7777
Call-ID: faif9ahhs9dd8==-sd98ajzz@host.example.com
CSeq: 1 ACK
```

The ACK is forwarded to the called party (message 6).

```
ACK sips:B@pc.example.com
Via: SIP/2.0/TLS app.example.com;branch=z9hG4bKh7s
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz9
From: Caller <sip:A@example.com>;tag=kkaz-
```



```
To: Callee <sip:B@example.com>;tag=7777
Call-ID: faif9ahhs9dd8==-sd98ajzz@host.example.com
CSeq: 1 ACK
```

Now, the application decides to push a user interface component to user A. So, it sends it a REFER request (message 7):

```
REFER sips:bad998asd8asd0000a@example.com SIP/2.0
Refer-To: https://app.example.com/script.pl
 ;context="kkaz-,7777,faif9ahhs9dd8==-sd98ajzz@host.example.com"
Via: SIP/2.0/TLS app.example.com;branch=z9hG4bK9zh6
Max-Forwards: 70
From: Recorder Application <sip:app.example.com>;tag=jhgf
To: Caller <sip:A@example.com>
Call-ID: 66676776767@app.example.com
CSeq: 1 REFER
Event: refer
Contact: <sips:app.example.com>
```

The REFER is answered by a 200 OK (message 8).

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS app.example.com;branch=z9hG4bK9zh6
From: Recorder Application <sip:app.example.com>;tag=jhgf
To: Caller <sip:A@example.com>;tag=pqoew
Call-ID: 66676776767@app.example.com
Supported: gruu, refer-context
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Contact: <sips:bad998asd8asd0000a@example.com>;schemes="http,sip,sips"
CSeq: 1 REFER
```

User A sends a NOTIFY (message 9):

```
NOTIFY sips:app.example.com SIP/2.0
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9320394238995
To: Recorder Application <sip:app.example.com>;tag=jhgf
From: Caller <sip:A@example.com>;tag=pqoew
Call-ID: 66676776767@app.example.com
CSeq: 1 NOTIFY
Max-Forwards: 70
Event: refer;id=93809824
Subscription-State: active;expires=3600
Contact: <sips:bad998asd8asd0000a@example.com>;schemes="http,sip,sips"
Content-Type: message/sipfrag;version=2.0
Content-Length: 20
```

SIP/2.0 100 Trying

And the recording server responds with a 200 OK (message 10)

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9320394238995
To: Recorder Application <sip:app.example.com>;tag=jhgf
From: Caller <sip:A@example.com>;tag=pqoew
Call-ID: 66676776767@app.example.com
CSeq: 1 NOTIFY
```

The REFER request contained a "context" Refer-To header field parameter with a valid dialog identifier. Furthermore, all of the signaling was over TLS and the dialog identifiers contain sufficient randomness. As such, the caller, A, automatically authorizes the application. It then acts on the Refer-To URI, fetching the script from app.example.com (message 11). The response, message 12, contains a web application that the user can click on to enable recording. Because the client executed the URL in the Refer-To, it generates another NOTIFY to the application, informing it of the successful response (message 13). This is answered with a 200 OK (message 14). When the user clicks on the link (message 15), the results are posted to the server, and an updated display is provided (message 16).

12. Security Considerations

There are many security considerations associated with this framework. It allows applications in the network to instantiate user interface components on a client device. Such instantiations need to be from authenticated applications, and also need to be authorized to place a UI into the client. Indeed, the stronger requirement is authorization. It is not so important to know that name of the provider of the application, but rather, that the provider is authorized to instantiate components.

This specification defines specific authorization techniques and requirements. Automatic authorization is granted if the application can prove that it is on the call path, or is trusted by an element on the call path. As documented above, this can be accompished by the use of cryptographically random dialog identifiers and the usage of sips for message confidentiality. It is RECOMMENDED that sips be implemented by user agents compliant to this specification. This does not represent a change from the requirements in RFC 3261.

13. IANA Considerations

13.1 SIP Option Tag

This specification registers a new SIP option tag, as per the guidelines in Section 27.1 of RFC 3261 [1].
Name: refer-context
Description: This option tag is used to identify the REFER extension that defines the "context" parameter of the Refer-To header field.

13.2 Header Field Parameter

This specification defines a new header field parameter, as per the registry created by [9]. The required information is as follows:
Header field in which the parameter can appear: Refer-To
Name of the Parameter context
RFC Reference RFC XXXX [[NOTE TO IANA: Please replace XXXX with the RFC number of this specification.]]

14. Contributors

This document was produced as a result of discussions amongst the application interaction design team. All members of this team contributed significantly to the ideas embodied in this document. The members of this team were:

Eric Burger
Cullen Jennings
Robert Fairlie-Cuninghame

15. References

15.1 Normative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

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

[4] McGlashan, S., Lucas, B., Porter, B., Rehor, K., Burnett, D., Carter, J., Ferrans, J. and A. Hunt, "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C CR CR-voicexml20-20030220, February 2003.

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

[6] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[7] Burger, E., "A Session Initiation Protocol (SIP) Event Package for Key Press Stimulus (KPML)", draft-ietf-sipping-kpml-03 (work in progress), May 2004.

[8] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-01 (work in progress), February 2004.

[9] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", draft-ietf-sip-parameter-registry-02 (work in progress), June 2004.

15.2 Informative References

[10] Peterson, J., "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-02 (work in progress), May 2004.

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

[12] Jennings, C., Peterson, J. and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002.

[13] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-01 (work in progress), October 2003.

[14] Rosenberg, J., Schulzrinne, H. and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", draft-ietf-sip-callerprefs-10 (work in progress), October 2003.

[15] Rosenberg, J. and H. Schulzrinne, "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-dialog-package-04 (work in progress), February 2004.

[16] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[17] Schulzrinne, H. and S. Petrack, "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals", RFC 2833, May 2000.

[18] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[19] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations", RFC 3680, March 2004.

Author's Address

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

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 procedures with respect to rights in RFC 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.

SIPPING Working Group A. Johnston
Internet-Draft MCI
Expires: January 16, 2005 O. Levin
Microsoft
July 18, 2004

Session Initiation Protocol Call Control - Conferencing for User Agents
draft-ietf-sipping-cc-conferencing-04

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 16, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This specification defines conferencing call control features for the Session Initiation Protocol (SIP). This document builds on the Conferencing Requirements and Framework documents to define how a tightly coupled SIP conference works. The approach is explored from different user agent (UA) types perspective: conference-unaware, conference-aware and focus UAs. The use of URIs in conferencing, OPTIONS for capabilities discovery, and call control using REFER are covered in detail with example call flow diagrams. The usage of the isfocus feature tag is defined. A SIP option tag for conference-aware UAs is defined.

Table of Contents

1. Introduction

This specification uses the concepts and definitions from the high level requirements [10] and the SIP conferencing framework [11] documents.

The approach described in this document implements key functions in the conferencing framework using SIP primitives only. This allows for conducting simple conferences with defined functionalities using SIP mechanisms and conventions. Many other advanced functions can be implemented using additional means but they are not in the scope of this document.

This document presents the basic call control (dial-in and dial-out) conferencing building blocks from the UA perspective. Possible applications include ad-hoc conferences and scheduled conferences.

Note that a single conference can bridge participants having different capabilities and who potentially have joined the conference by different means (i.e. dial-in, dial-out, scheduled, and ad-hoc).

The call control and dialog manipulation approach is based on the multiparty framework [12] document. That document defines the basic approach of service design adopted for SIP which includes:

- Definition of primitives, not services
- Signaling model independent
- Invoker oriented
- Primitives make full use of URIs
- Include authentication, authorization, logging, etc. policies
- Define graceful fallback to baseline SIP.

The use of opaque URIs and the ability to communicate call control context information within a URI (as opposed to service-related header fields), as discussed in RFC 3087 [13], is fundamental to this approach.

Capabilities discovery is an important feature of SIP systems, and conferencing systems can make use of such features. For a UA acting as a focus in a conference, this specification defines the usage of the 'isfocus' feature parameter. For a UA which supports the extensions and scenarios described in this document, the 'conf' option tag is defined.

2. Usage of the 'isfocus' Feature Parameter

2.1 General

The main design guidelines for the development of SIP extensions and conventions for conferencing are to define the minimum number of extensions and to have seamless backwards compatibility with conference-unaware SIP UAs. The minimal requirement for SIP is being able to express that a dialog is a part of a certain conference referenced to by a URI. As a result of these extensions, it is possible to do the following using SIP:

- Create a conference
- Join a conference
- Invite a user to a conference
- Expel a user by third party
- Discover if a URI is a conference URI
- Delete a conference

The approach taken is to use the feature parameter "isfocus" to express that a SIP dialog belongs to a conference. The use of feature parameters in Contact header fields to describe the characteristics and capabilities of a UA is described in the User Agent Capabilities [7] document which includes the definition of the "isfocus" feature parameter.

2.2 Session Establishment

In session establishment, a focus MUST include the "isfocus" feature parameter in the Contact header field unless the focus wishes to hide the fact that it is a focus. To a participant, the feature parameter will be associated with the remote target URI of the dialog. It is an indication to a conference-aware UA that the resulting dialog belongs to a conference identified by the URI in the Contact header field and that the call control conventions defined in this document can be applied.

The Conference URI MUST meet the requirements to be a GRUU (Globally Routable User Agent URI) as detailed in [9]

2.3 OPTIONS

Currently the only met requirement is: given an opaque URI, being able to recognize whether it belongs to a certain conference (i.e. meaning that it is a conference URI) or not. As with any other OPTIONS request, it can be done either inside an active dialog or outside a dialog. A focus MUST include the "isfocus" feature parameter in a 200 OK response to an OPTIONS unless the focus wishes to hide the fact that it is a focus.

3. SIP User Agent Conferencing Capability Types

From a conferencing perspective, the framework document outlines a number of possible different SIP components such as conference-unaware participant, conference-aware participant, and focus.

This document applies the concepts above to the SIP call control part of the conferencing components. It defines normative behavior of the SIP UAs in various conferencing situations (referred later as "scenarios").

3.1 Focus UA

A focus, as defined in the framework, hosts a SIP conference and maintains a SIP signaling relationship with each participant in the conference. A focus contains a conference-aware user agent that supports the conferencing call control conventions as defined in this document.

A focus SHOULD support the conference package [5] and indicate so in Allow-Events header fields in requests and responses. A focus MAY include information about the conference in SDP message bodies sent.

A focus SHOULD support the Replaces [8] header field.

A user agent with focus capabilities could be implemented in end user equipment and would be used for the creation of ad-hoc conferences.

A dedicated conferencing server, whose primary task is to simultaneously host conferences of arbitrary type and size, may allocate and publish a conference factory URI (as defined in the next section) for creating an arbitrary number of ad-hoc conferences (and subsequently their focuses) using SIP call control means.

3.2 Conference Factory URI

According to the framework, there are many ways in which a conference can be created. These are open to the conferencing server implementation policy and include non-automated means (such as IVR), SIP, and a conference policy control protocol.

In order to automatically create an arbitrary number of ad-hoc conferences (and subsequently their focuses) using SIP call control means, a globally routable Conference Factory URI can be allocated and published.

A successful attempt to establish a call to this URI would result in the automatic creation a new conference and its focus. As a result, note that the Conference Factory URI and the newly created focus URI MAY resolve to different physical devices.

A scenario showing the use of the conference factory URI is shown in Section 4.5.

3.3 Conference-Unaware UA

The simplest user agent can participate in a conference ignoring all SIP conferencing-related information. The simplest user agent is able to dial into a conference and to be invited to a conference. Any conferencing information is optionally conveyed to/from it using non-SIP means. Such a user agent would not usually host a conference (at least, not using SIP explicitly). A conference-unaware UA needs only to support RFC 3261 [2]. Call flows for conference-unaware UAs are not shown in general in this document as they would be identical to those in the SIP call flows [15] document.

3.4 Conference-Aware UA

A conference-aware user agent supports SIP conferencing call control conventions defined in this document as a conference participant, in addition to support of RFC 3261.

A conference-aware UA MUST recognize the "isfocus" feature parameter. A conference-aware UA SHOULD support REFER [3], SIP events [4], and the conferencing package [5].

A conference-aware UA SHOULD subscribe to the conference package if the "isfocus" parameter is in the remote target URI of a dialog and if the conference package is listed by a focus in an Allow-Events header field. The SUBSCRIBE to the conference package should be sent outside any INVITE-initiated dialog. A termination of the INVITE dialog with a BYE does not necessarily terminate the SUBSCRIBE dialog.

A conference-aware UA MAY render to the user any information about the conference obtained from the SIP header fields and SDP fields from the focus.

This specification defines a new SIP option tag, "conf" for use by UAs. A conference-aware UA SHOULD include the "conf" option tag in a Supported header field in requests and responses. A conference-aware UA discovering support for the "conf" option tag MAY make use the scenarios described in this specification. Note that a UA which is only acting as a focus (i.e. is not acting as a combined focus/

participant) in a dialog SHOULD NOT indicates support for this specification using the "conf" option tag. Instead, the "isfocus" feature tag should be used.

4. SIP Conferencing Primitives

The SIP conferencing call control flows presented in this section are the call control building blocks for various SIP tight conferencing applications as described in the conferencing requirements [10] and framework [11] documents. The major design goal is that the same SIP conferencing primitives would be used by user agents having different conferencing capabilities and comprising different applications.

4.1 Joining a Conference using the Conference URI - Dial In

In this section a user knows the conference URI and "dials in" to join this conference.

If the UA is the first participant of the conference to dial in, it is likely that this INVITE will create the focus and hence the conference. However, the conference URI must have been reserved prior to its use.

If the conference is up and running already, the dialing-in participant is joined to the conference by its focus.

To join an existing specific conference a UA SHOULD send an INVITE with the Request-URI set to the conference URI. The focus MUST include the "isfocus" feature parameter in the Contact header field of the 200 OK response to the INVITE.

An example call flow for joining a conference is shown in Figure 1.

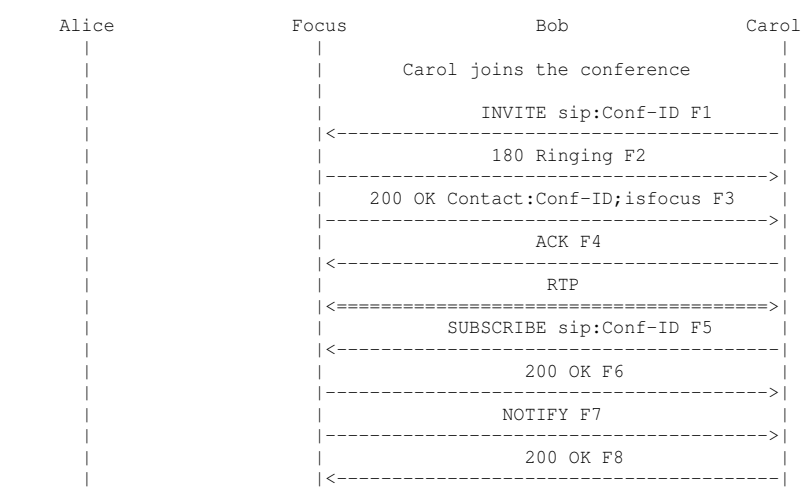


Figure 1. A Participant Joins a Conference using the Conference URI.

```
 F1   INVITE sip:3402934234@example.com SIP/2.0
      Via: SIP/2.0/UDP client.chicago.example.com
       ;branch=z9hG4bKhjhs8ass83
      Max-Forwards: 70
      To: <sip:3402934234@example.com>
      From: Carol <sip:carol@chicago.example.com>;tag=32331
      Call-ID: d432fa84b4c76e66710
      CSeq: 45 INVITE
      Contact: <sip:carol@client.chicago.example.com>
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
       SUBSCRIBE, NOTIFY
      Allow-Events: dialog
      Accept: application/sdp, message/sipfrag
      Supported: conf, replaces
      Content-Type: application/sdp
      Content-Length: 274

      (SDP not shown)


 F3   SIP/2.0 200 OK
```



```
      Via: SIP/2.0/UDP client.chicago.example.com
       ;branch=z9hG4bKhjhs8ass83;received=192.0.2.4
      To: <sip:3402934234@example.com>;tag=733413
      From: Carol <sip:carol@chicago.example.com>;tag=32331
      Call-ID: d432fa84b4c76e66710
      CSeq: 45 INVITE
      Contact: <sip:3402934234@example.com>;isfocus
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
       SUBSCRIBE, NOTIFY
      Allow-Events: dialog, conference
      Accept: application/sdp, application/conference-info+xml,
       message/sipfrag
      Supported: replaces, join, gruu
      Content-Type: application/sdp
      Content-Length: 274

      v=0
      o=focus431 2890844526 2890842807 IN IP4 ms5.conf.example.com
      s=Example Subject
      i=Example Conference Hosted by Example.com
      u=http://conf.example.com/3402934234
      e=3402934234@conf-help.example.com
      p=+1-888-2934234
      c=IN IP4 ms5.conf.example.com
      t=0 0
      m=audio 49170 RTP/AVP 0
      m=video 51372 RTP/AVP 31


 F5   SUBSCRIBE sip:3402934234@example.com SIP/2.0
      Via: SIP/2.0/UDP client.chicago.example.com
       ;branch=z9hG4bKdf334
      Max-Forwards: 70
      To: <sip:3402934234@example.com>
      From: Carol <sip:carol@chicago.example.com>;tag=43524545
      Call-ID: k3l43id034ksereree
      CSeq: 22 SUBSCRIBE
      Contact: <sip:carol@chicago.example.com>
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
       SUBSCRIBE, NOTIFY
      Event: conference
      Accept: application/sdp, message/sipfrag
      Supported: conf, replaces
      Content-Length: 0


 F7   NOTIFY sip:carol@chicago.example.com SIP/2.0
```

```
Via: SIP/2.0/UDP ms5.conf.example.com;branch=z9hG4bK3343d1
Max-Forwards: 70
To: Carol <sip:carol@chicago.example.com>;tag=43524545
From: <sip:3402934234@example.com>;tag=a3343df32
Call-ID: k3l43id034kseреree
CSeq: 34321 NOTIFY
Contact: <sip:3402934234@example.com>;isfocus
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
 SUBSCRIBE, NOTIFY
Event: conference
Accept: application/sdp, message/sipfrag
Subscription-State: active;expires=3600
Supported: replaces, join, gruu
Content-Type: application/conference-info+xml
Content-Length: ...

<conference-info version="0" state="full"
             entity="sip:3402934234@example.com">
  <user uri="sip:carol@chicago.example.com"
                           display-name="Carol">
    <status>connected</status>
    <joining-mode>dialed-in</joining-mode>
    <media-stream media-type="audio">
       <proto>RTP/AVP</proto>
       <ssrc>583398</ssrc>
    </media-stream>
    <media-stream media-type="video">
       <proto>RTP/AVP</proto>
       <ssrc>345212</ssrc>
    </media-stream>
  </user>
  <conf-uri>tel:+18882934234</conf-uri>
</conference-info>
```

4.2 Adding a Participant by the Focus - Dial Out

To directly add a participant to a conference, a focus SHOULD send an INVITE to the participant containing a Contact header field with the conference URI and the "isfocus" feature parameter.

Note that a conference-unaware UA would simply ignore the conferencing information and treat the session (from a SIP perspective) as a point to point session.

An example call flow is shown in Figure 2. It is assumed that Alice is already a participant of the conference. The focus invites Carol

to the conference by sending an INVITE. After the session is established, Carol subscribes to the conference URI. It is important to note that there is no dependency on Carol's SUBSCRIBE (F5) and the NOTIFY to Alice (F9) - they occur asynchronously and independently.

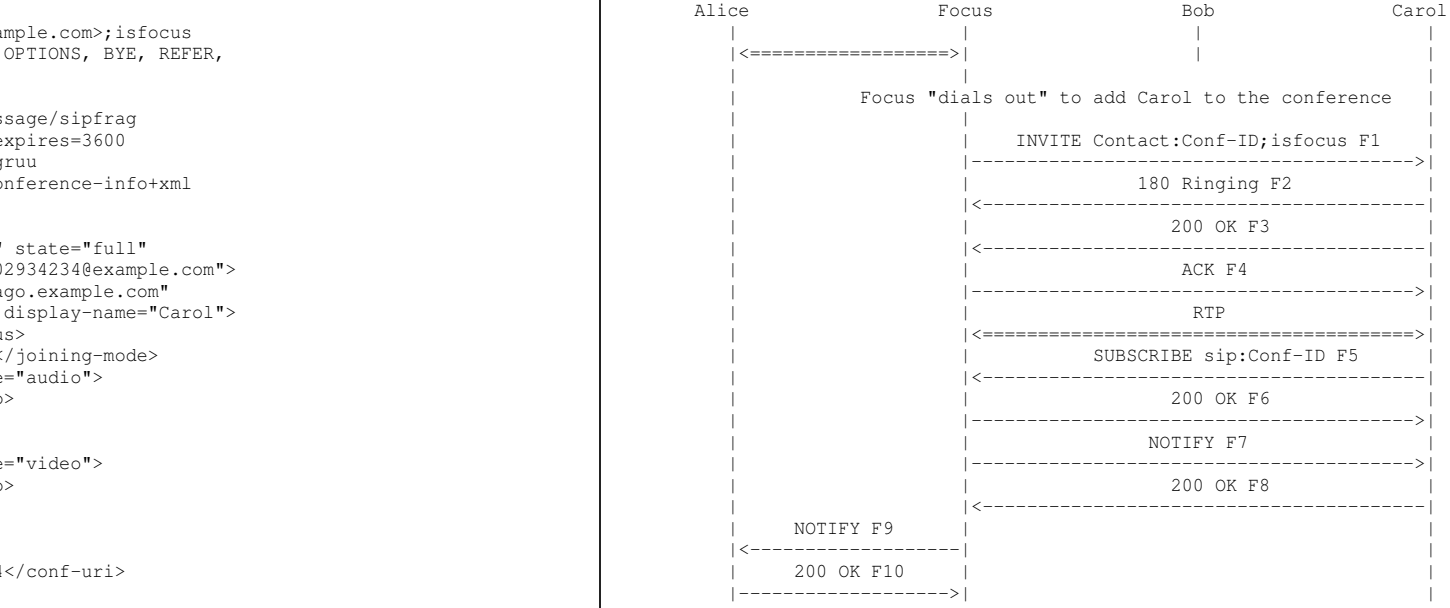


Figure 2. A Focus "dials out" to Add a Participant to the Conference.

```
F7    NOTIFY sip:carol@chicago.example.com SIP/2.0
      Via: SIP/2.0/UDP ms5.conf.example.com;branch=z9hG4bK3343d1
      Max-Forwards: 70
      To: Carol <sip:carol@chicago.example.com>;tag=43524545
      From: <sip:3402934234@example.com>;tag=a3343df32
      Call-ID: k3l43id034kseреree
      CSeq: 34321 NOTIFY
      Contact: <sip:3402934234@example.com>;isfocus
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
       SUBSCRIBE, NOTIFY
```

```
Event: conference
Accept: application/sdp, message/sipfrag
Subscription-State: active;expires=3600
Supported: replaces, gruu
Content-Type: application/conference-info+xml
Content-Length: ...

<conference-info version="0" state="full"
             entity="sip:3402934234@example.com">
  <user uri="sip:alice@atlanta.example.com"
                            display-name="Alice">
    <status>connected</status>
    <joining-mode>dialed-in</joining-mode>
    <media-stream media-type="audio">
       <proto>RTP/AVP</proto>
       <ssrc>674231</ssrc>
    </media-stream>
    <media-stream media-type="video">
       <proto>RTP/AVP</proto>
       <ssrc>213563</ssrc>
    </media-stream>
  </user>
  <user uri="sip:carol@chicago.example.com"
                            display-name="Carol">
    <status>connected</status>
    <joining-mode>dialed-out</joining-mode>
    <media-stream media-type="audio">
       <proto>RTP/AVP</proto>
       <ssrc>583398</ssrc>
    </media-stream>
    <media-stream media-type="video">
       <proto>RTP/AVP</proto>
       <ssrc>345212</ssrc>
    </media-stream>
  </user>
  <conf-uri>tel:+18882934234</conf-uri>
</conference-info>
```

F9

```
NOTIFY sip:alice@atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP ms5.conf.example.com;branch=z9hG4bK3432
Max-Forwards: 70
To: Alice <sip:alice@atlanta.example.com>;tag=43524545
From: <sip:3402934234@example.com>;tag=a3343df32
Call-ID: 8820450524545
CSeq: 998 NOTIFY
Contact: <sip:3402934234@example.com>;isfocus
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
```



```
 SUBSCRIBE, NOTIFY
Event: conference
Accept: application/sdp, message/sipfrag
Subscription-State: active;expires=2450
Supported: replaces, gruu
Content-Type: application/conference-info+xml
Content-Length: ...

<conference-info version="0" state="partial"
               entity="sip:3402934234@example.com">
  <user uri="sip:carol@chicago.example.com"
                                 display-name="Carol">
    <status>connected</status>
    <joining-mode>dialed-out</joining-mode>
    <media-stream media-type="audio">
       <proto>RTP/AVP</proto>
       <ssrc>583398</ssrc>
    </media-stream>
    <media-stream media-type="video">
       <proto>RTP/AVP</proto>
       <ssrc>345212</ssrc>
    </media-stream>
  </user>
  <conf-uri>tel:+18882934234</conf-uri>
</conference-info>
```

4.3 Manually Creating a Conference by Dialing into a Conferencing Application

In this section, a user sends an INVITE to a conference server application. The application (such as an IVR system or a web page) is implemented because the system requires additional input from the user before it is able to create a conference. After a normal dialog is established, additional information is received and the conference together with its focus are created. At this point the conference server MUST re-INVITE the user with the conference URI in Contact with the "isfocus" feature parameter.

Alternatively, the additional information MAY be provided by the user during an early dialog established. This could be accomplished by a 183 Session Progress response sent by the conferencing application. After the conference is created, the conference URI MUST then be returned in a Contact in the 200 OK.

An example call flow is shown in Figure 3. In this example, Alice

uses a conference application which is triggered when Alice sends an INVITE to the conference application. In this example, Conf-App is used to represent the conference application URI. Alice's conference-aware UA learns of the existence of the conference from the "isfocus" feature parameter and subscribes to the conference package to receive notifications of the conference state.

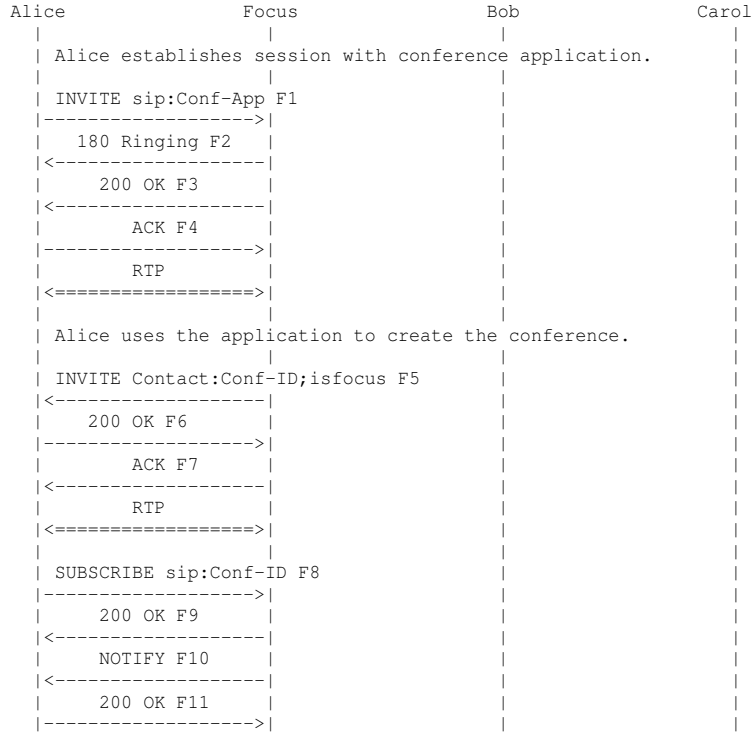


Figure 3. A Participant Creates a Conference using an Application.

4.4 Creating a Conference using Ad-Hoc SIP Methods

This section addresses creating a conference by using ad-hoc SIP

means. The conference factory URI (as defined in Section 2.4) is used to automatically create the conference in this example.

The benefit of this approach is that the conference URI need not be known to the user - instead it is created by a focus and used by the participants' UAs. The main difference between this scenario and Section 4.3 is that no user intervention (IVR, web page form, etc.) is required to create the conference.

The SIP URI of the conference factory can be provisioned in the UA (as in a "create new conference" button on a SIP phone) or can be discovered using other means.

A SIP entity (such as conferencing server) can distinguish this INVITE request as a request to create a new ad-hoc conference from a request to join an existing conference by the Request-URI.

Assuming that all security and policy requirements have been met, a new conference will be created with the Contact URI returned in the 200 OK being the conference URI. The Contact header field MUST contain the "isfocus" feature parameter to indicate that this URI is for a conference.

An example call flow is shown in Figure 4. Note that Conf-Factory is shorthand for the conference factory URI and Conf-ID Is short for the conference URI. In this flow, Alice has a conference-aware UA and creates a conference by sending an INVITE to the conference factory URI. The conference factory application creates the conference and redirects using a 302 Moved Temporarily response to the focus. Note that with proxy recursion, Alice may never see the redirect but may just receive the responses from the focus starting with message F5. Once the media session is established, Alice subscribes to the conference URI obtained through the Contact in the 200 OK response from the focus.

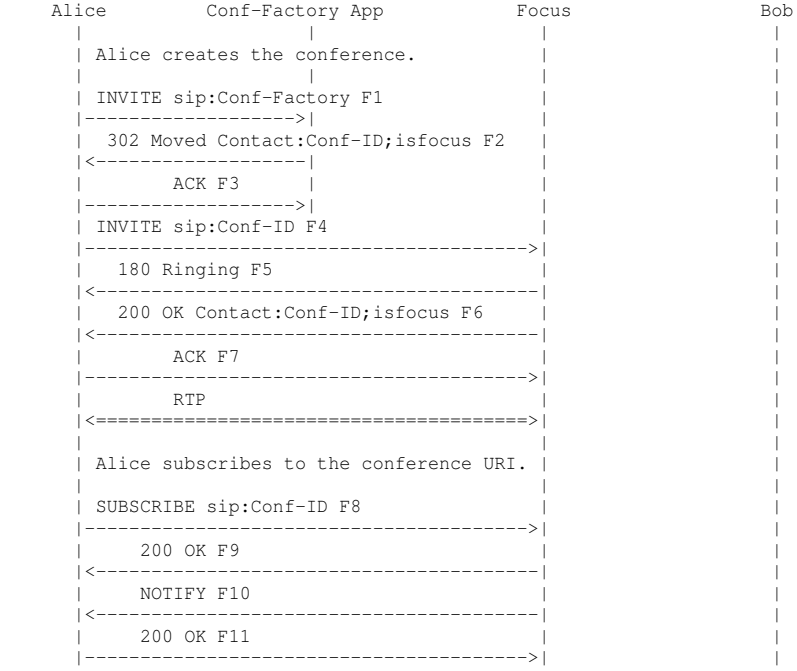


Figure 4. Creation of a Conference using SIP Ad-Hoc Methods.

4.5 Requesting the Focus Add a New Resource to a Conference

A SIP conference URI can be used to inject different kinds of information into the conference. Examples include new participants, new real-time media sources, new IM messages, and pointers to passive information references (such as HTTP URIs).

To request the focus add a new information resource to the specified conference, any SIP UA can send a REFER to the conference URI with a Refer-To containing the URI of the new resource. Since this REFER is sent to the conference URI and not the conference factory URI, the semantics to the focus are to bring the resource into the conference and make it visible to the conference participants. The resultant

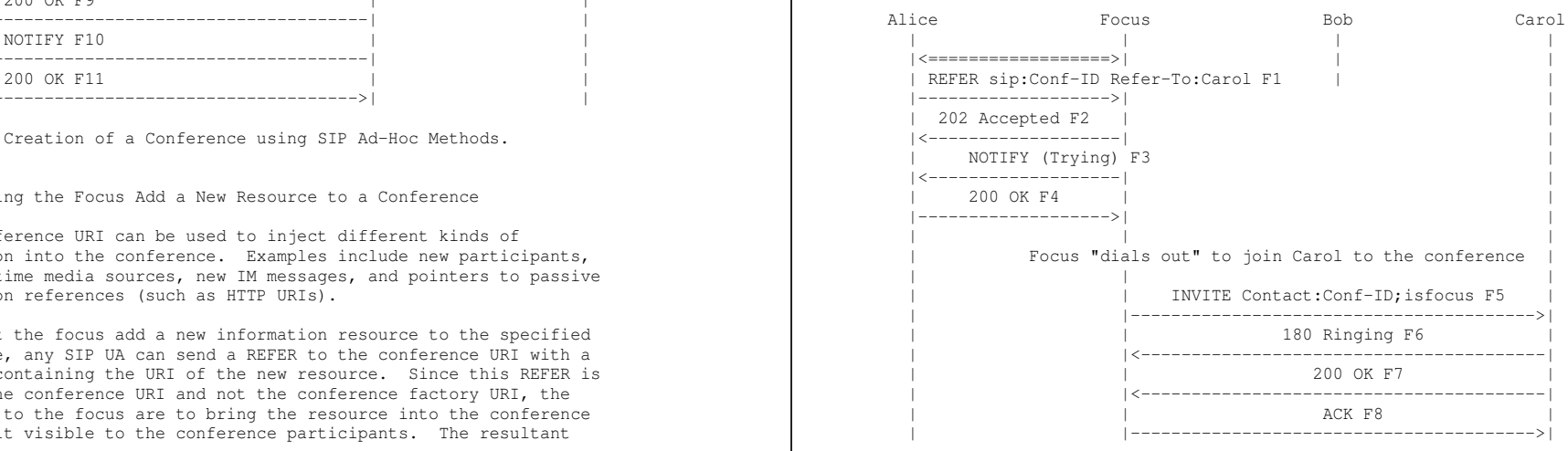
focus procedures are dependant both on the nature of the new resource (as expressed by its URI) and the own focus policies regarding IM, central vs. distributed real time media processing, etc.

The scenario for adding a new UA participant is important to support because it works even if the new participant does not support REFER and transfer call control - only the requesting participant and the focus need to support the REFER and transfer call control.

Upon receipt of the REFER containing a Refer-To header with a SIP URI, the focus SHOULD send an INVITE to the new participant identified by the Refer-To SIP URI containing a Contact header field with the conference URI and the "isfocus" feature parameter.

A conference-unaware UA would simply ignore the conferencing information and treat the session (from a SIP perspective) as a point to point session.

An example call flow is shown in Figure 5. It is assumed that Alice is already a participant of the conference. Alice sends a REFER to the conference URI. The focus invites Carol to the conference by sending an INVITE. After the session is established, Carol subscribes to the conference URI. It is important to note that there is no dependency on Carol's SUBSCRIBE (F11) and the NOTIFY to Alice (F15) - they occur asynchronously and independently.



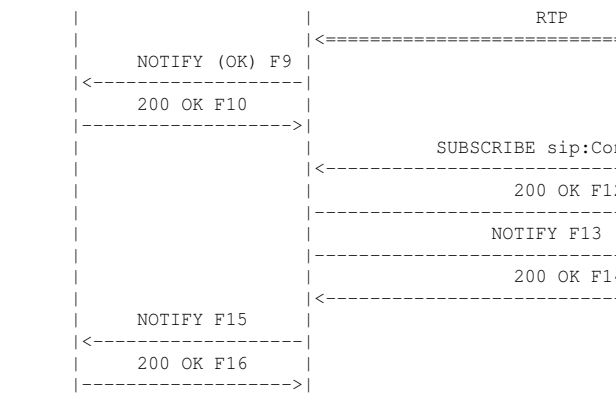


Figure 5. Participant Requests Focus add a Participant to the Conference.

```
 F1   REFER sip:3402934234@example.com SIP/2.0
      Via: SIP/2.0/UDP client.atlanta.example.com;branch=z9hG4bKg45344
      Max-Forwards: 70
      To: <sip:3402934234@example.com>
      From: Alice <sip:alice@atlanta.example.com>;tag=5534562
      Call-ID: 849392fklgl43
      CSeq: 476 REFER
      Contact: <sip:alice@alice.example.com>
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
       SUBSCRIBE, NOTIFY
      Accept: application/sdp, message/sipfrag
      Refer-To: <sip:carol@chicago.example.com>
      Supported: conf, replaces
      Content-Length: 0
```

4.6 Adding a 3rd Party Using Conference URI

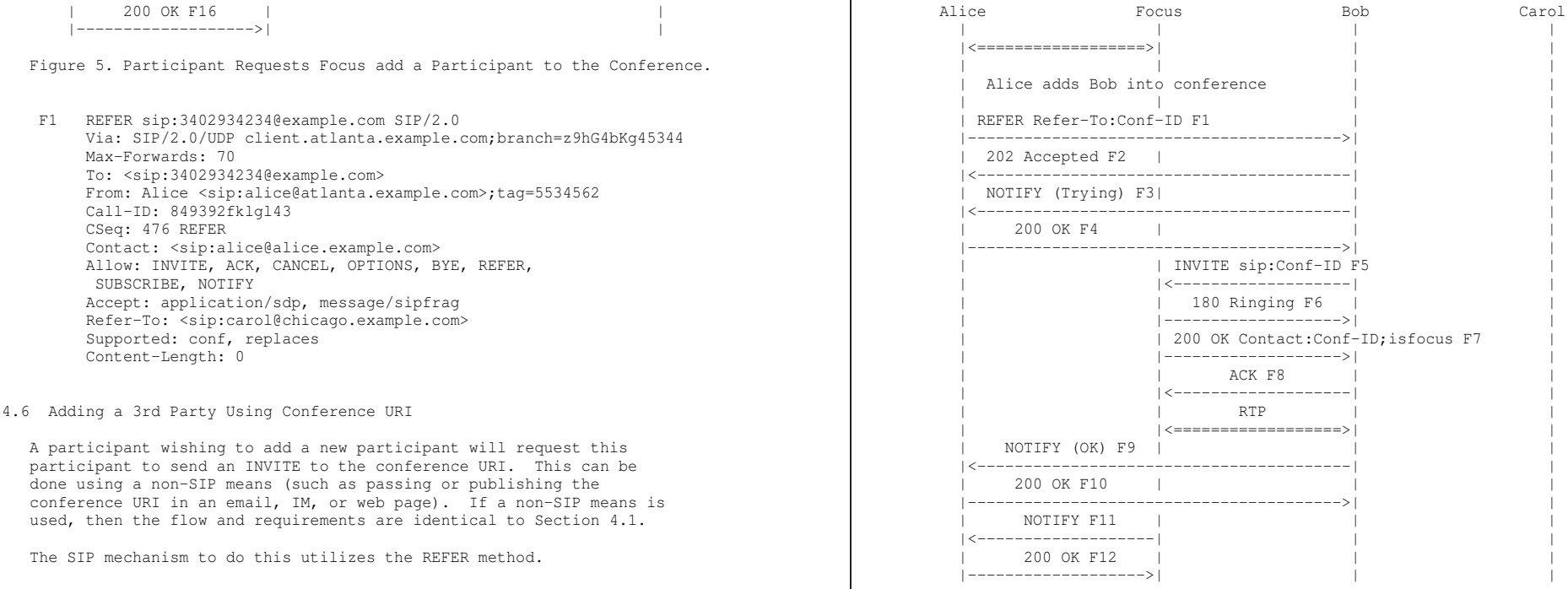
A participant wishing to add a new participant will request this participant to send an INVITE to the conference URI. This can be done using a non-SIP means (such as passing or publishing the conference URI in an email, IM, or web page). If a non-SIP means is used, then the flow and requirements are identical to Section 4.1.

The SIP mechanism to do this utilizes the REFER method.

A UA wishing to add a new participant SHOULD send a REFER request to the participant with a Refer-To header containing the conference URI.

The requirements are then identical to the "dial in" case of Section 4.1. The inviting participant MAY receive notification through the REFER action that the new participant has been added in addition to the notification received through the conference package.

An example is shown in Figure 7. In this call flow, it is assumed that Alice is already a participant of the conference. Alice sends Bob an "out of band" REFER - that is, a REFER outside of an established dialog. Should Bob reject the REFER, Alice might try sending an INVITE to Bob to establish a session first, then send a REFER within the dialog, effectively transferring Bob into the conference [17].



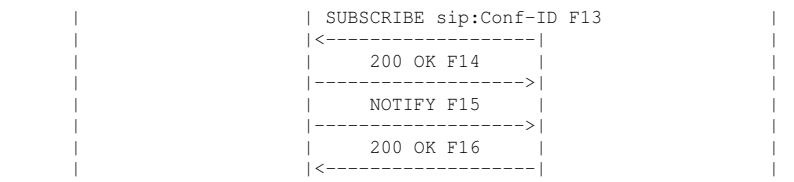


Figure 6. Adding a Participant to an Existing Conference.

4.7 Requesting Focus Refer a Participant into the Conference

A participant may request the focus refer a participant into the conference by sending a REFER method. The Refer-To header field will have the method set to REFER and an escaped Refer-To header field containing the conference URI.

Note that in Message F1 below, the Refer-To header field is shown as continuing across two lines - this would not be the case in an actual message, the URI would have continued beyond the formatting limitations of this document.

This scenario is shown in Figure 7.

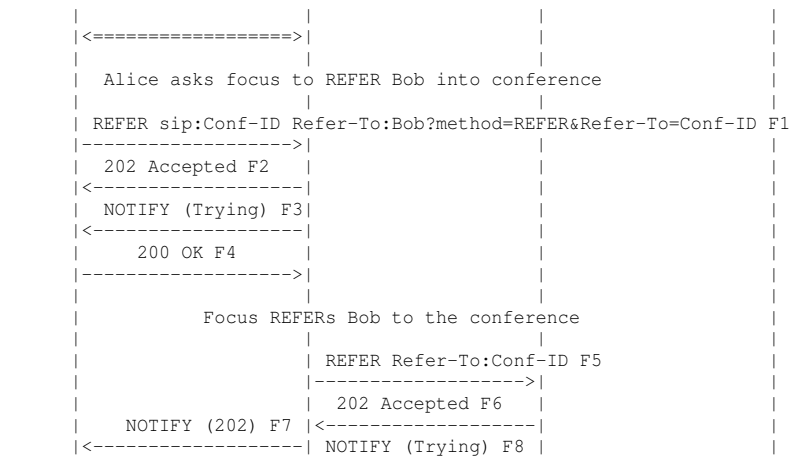


Figure 7. Requesting a Focus Refer a Participant to a Conference.

```
F1    REFER sip:3402934234@example.com SIP/2.0
      Via: SIP/2.0/UDP client.atlanta.example.com;branch=z9hG4bKg45344
      Max-Forwards: 70
      To: <sip:3402934234@example.com>
      From: Alice <sip:alice@atlanta.example.com>;tag=5534562
      Call-ID: 849392fklgl43
      CSeq: 476 REFER
      Contact: <sip:alice@alice.example.com>
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
       SUBSCRIBE, NOTIFY
      Accept: application/sdp, message/sipfrag
      Refer-To: <sip:bob@biloxi.example.com?method=REFER
                                  &Refer-To=sip:3402934234%40example.com>
      Supported: conf, replaces
      Content-Length: 0


F5    REFER sip:3402934234@example.com SIP/2.0
      Via: SIP/2.0/UDP ms5.conf.example.com;branch=z9hG4bK33445243
```

```
Max-Forwards: 70
To: <sip:bob@biloxi.example.com>
From: <sip:3402934234@example.com>;tag=345621412
Call-ID: 5494204
CSeq: 4524323 REFER
Contact: <sip:3402934234@example.com>;isfocus
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
 SUBSCRIBE, NOTIFY
Accept: application/sdp, message/sipfrag
Refer-To: <sip:3402934234@example.com>
Supported: join, gruu, replaces
Content-Length: 0
```

F11

```
INVITE sip:3402934234@example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.com;branch=z9hG4bKh3887
Max-Forwards: 70
To: <sip:3402934234@example.com>
From: Bob <sip:bob@biloxi.example.com>;tag=32411
Call-ID: 5d4324fa84b4c76e66710
CSeq: 764 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
 SUBSCRIBE, NOTIFY
Allow-Events: dialog
Accept: application/sdp, message/sipfrag
Supported: conf, replaces, join
Content-Type: application/sdp
Content-Length: 274

(SDP not shown)
```

4.8 Adding a 3rd Party Using a Dialog Identifier

Under some circumstances, a participant wanting to join a conference may only know a dialog identifier of one of the legs of the conference. The information may have been learned using the dialog package [18] or some non-SIP means to retrieve this information from a conference participant.

A UA can request to be added to a conference by sending a request to the focus containing a Join [6] header field containing a dialog ID of one leg of the conference (a dialog between a participant and the focus).

There are other scenarios in which a UA can use the Join header for

certain conferencing call control scenarios. See [6] for further examples and details.

An example is shown in Figure 8. It is assumed that Alice is a participant of the conference. The dialog identifier between Alice and the focus is abbreviated as A-F and is known by Bob. Bob requests to be added to the conference by sending an INVITE message F1 to the focus containing a Join header which contains the dialog identifier A-F. Bob is added into the conference by the focus.

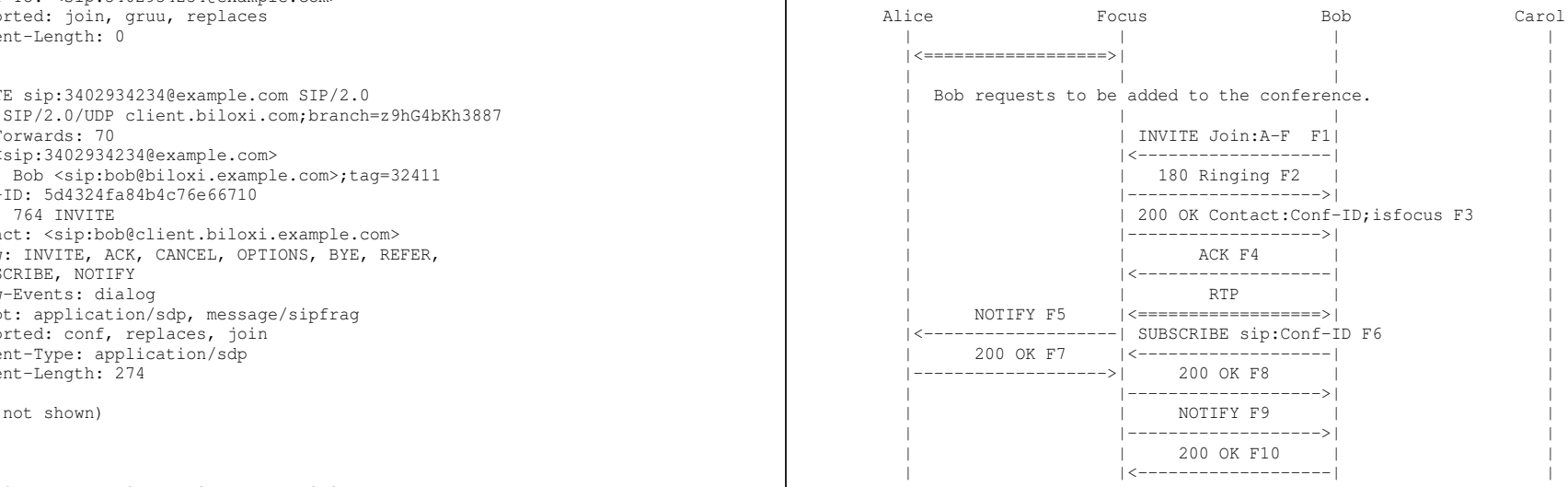


Figure 8. Adding a Participant to an Existing Conference using Join.

```
F1   INVITE sip:3402934234@example.com SIP/2.0
     Via: SIP/2.0/UDP client.biloxi.com;branch=z9hG4bKh3832
     Max-Forwards: 70
     To: <sip:3402934234@example.com>
     From: Bob <sip:bob@biloxi.example.com>;tag=32411
     Call-ID: d432fa84b4c76e66710
     CSeq: 8 INVITE
     Contact: <sip:bob@client.biloxi.example.com>
     Join: 3434034-293553453;to-tag=fdj3l34;from-tag=12f331
     Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
      SUBSCRIBE, NOTIFY
     Allow-Events: dialog
     Accept: application/sdp, message/sipfrag
     Supported: conf, replaces, join
     Content-Type: application/sdp
     Content-Length: 274

     (SDP not shown)
```

4.9 Changing User Agents within a Conference

A participant in a conference may want to change the user agent with which they participate in the conference. While this could be done by simply sending a BYE from one user agent to leave the conference and an INVITE from the other user agent to rejoin. However, the SIP Replaces [6] primitive is perfectly suited to this operation.

An example is shown in Figure 9. It is assumed that Alice is a participant of the conference using user agent #1. The dialog identifier between Alice's user agent #1 and the focus is abbreviated as A-F. Alice switches to user agent #2 and sends an INVITE message F1 to the focus containing a Replaces header which contains the dialog identifier A-F. Note that this dialog identifier could be learned through some non-SIP mechanism, or by use of SUBSCRIBE/NOTIFY and the dialog event package [18]. Alice's user agent #2 is added into the conference by the focus. The focus sends a BYE to user agent #1. User agent #1 then automatically terminates the subscription by sending a SUBSCRIBE with Expires:0 to terminate the subscription. Note that as the participant list (roster) has not changed during this scenario, no NOTIFYs are sent by the focus to subscribers to the participant list.

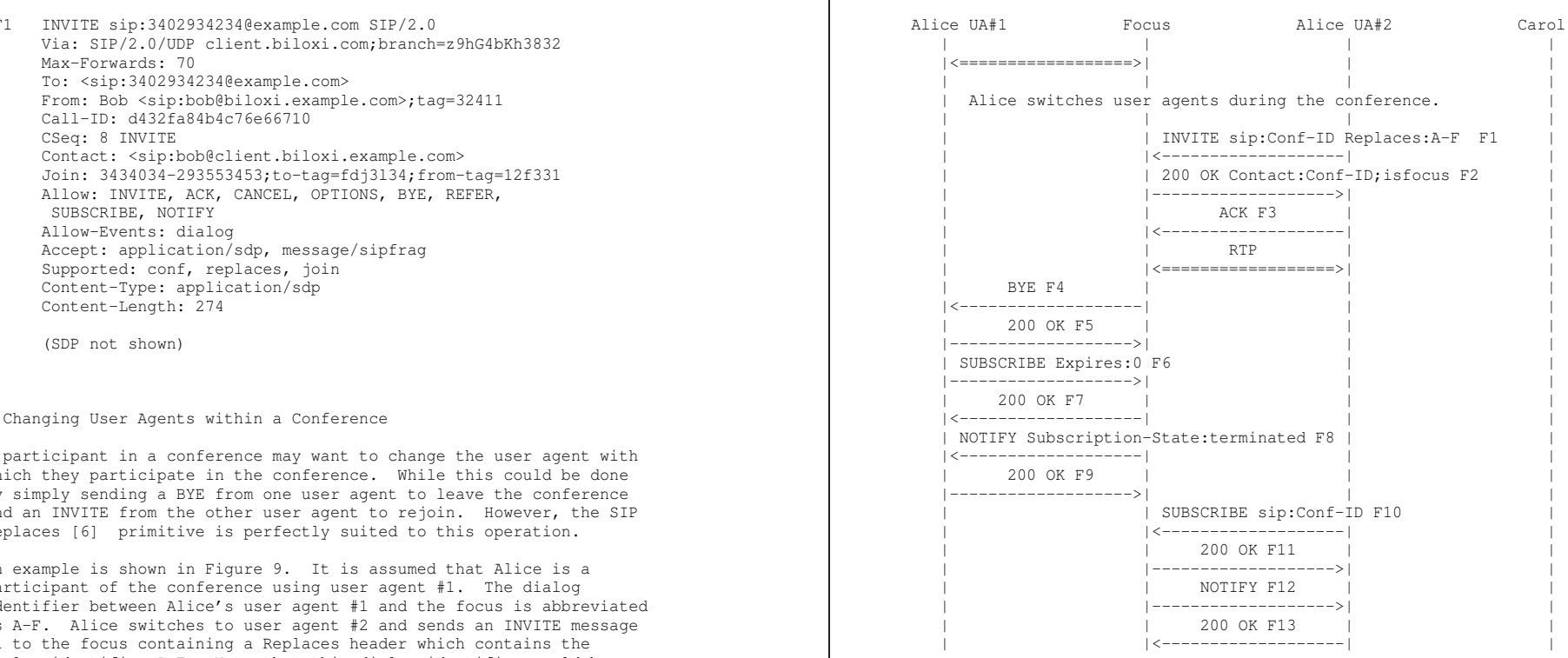


Figure 9. Adding a Participant to an Existing Conference using Join.

4.10 Bringing a Point-to-Point Session into a Conference

This call flow shows how a point to point call can be switched to a conference call involving an external focus.

Alice and Bob have an established session with a dialog identifier A-B. Alice joins the conference with the focus by sending an INVITE to the Conference URI. Alice then sends a REFER to the focus and includes an escaped Replaces header field in the Refer-To header field. Bob receives the INVITE from the focus, matches the dialog in

the Replaces header field with the dialog with Alice. As a result, Bob accepts the INVITE, joins the conference, and sends a BYE to Alice to tear down their point to point dialog.

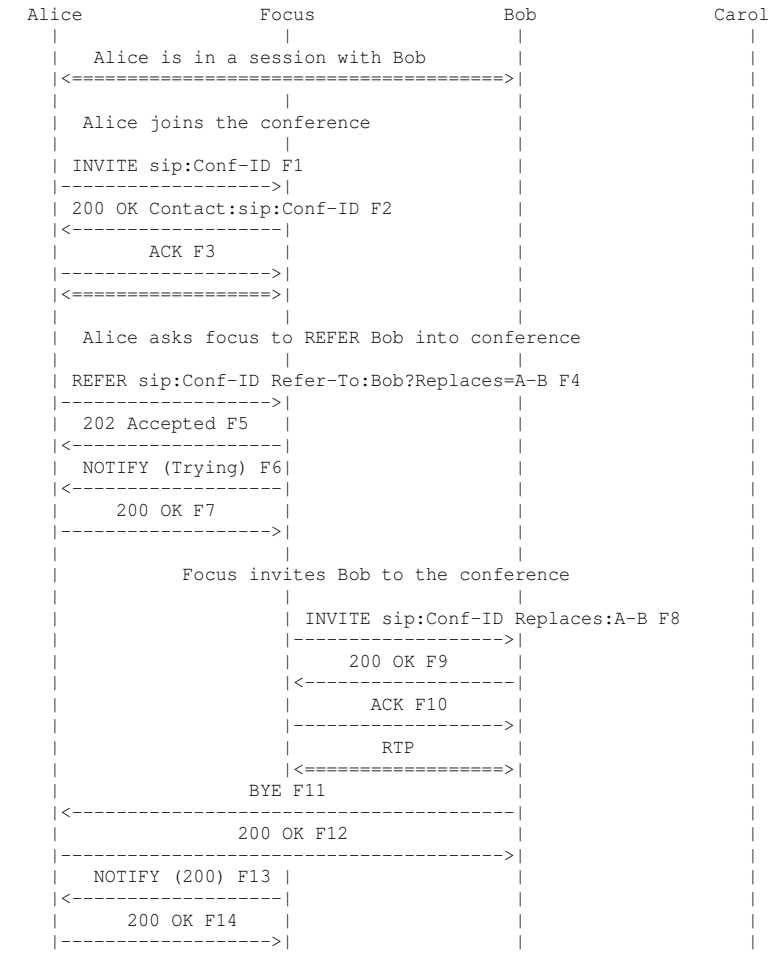


Figure 10. Transitioning a Point to Point Session into a Conference.

4.11 Requesting the Focus Remove a Participant from a Conference

To request the focus remove a participant from the specified conference, a properly authorized SIP UA (typically the conference owner) can send a REFER to the conference URI with a Refer-To containing the URI of the participant and with the method set to BYE. The requestor does not need to know the dialog information about the dialog between the focus and the participant who will be removed - the focus knows this information and fills it when it generates the BYE request.

An example call flow is shown in Figure 11. It is assumed that Alice and Carol are already participants of the conference and that Alice is authorized to remove members from the conference. Alice sends a REFER to the conference URI with a Refer-To header containing a URI of the form &ltsip:carol@chicago.example.com?method=BYE>.

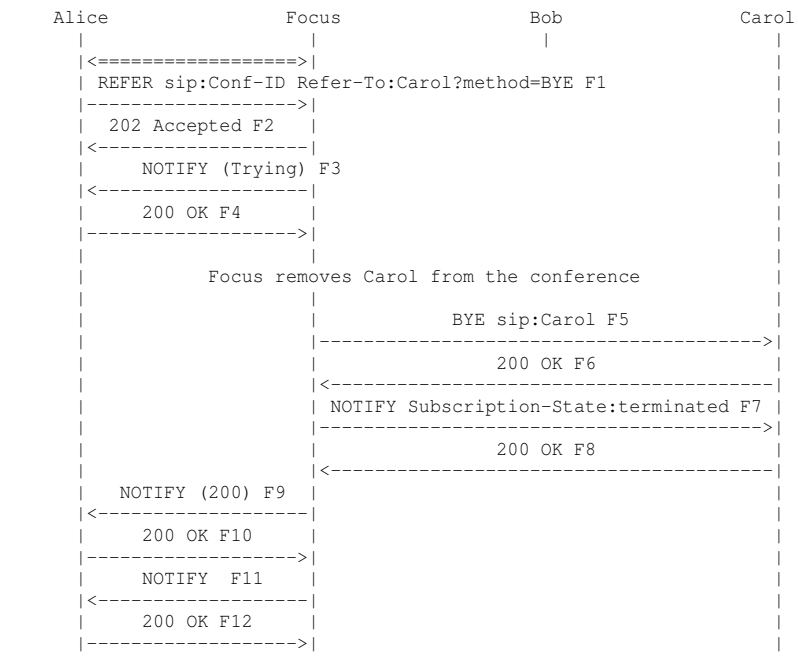


Figure 11. Participant Requests Focus Remove a Participant from the Conference.


```
F1   REFER sip:3402934234@example.com SIP/2.0
     Via: SIP/2.0/UDP client.atlanta.example.com;branch=z9hG4bKg45344
     Max-Forwards: 70
     To: <sip:3402934234@example.com>
     From: Alice <sip:alice@atlanta.example.com>;tag=5534562
     Call-ID: 849392fklgl43
     CSeq: 476 REFER
     Contact: <sip:alice@alice.example.com>
     Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
      SUBSCRIBE, NOTIFY
     Accept: application/sdp, message/sipfrag
     Refer-To: <sip:carol@chicago.example.com?method=BYE>
     Supported: conf, replaces
     Content-Length: 0


F5   BYE sip:carol@client.chicago.example.com SIP/2.0
     Via: SIP/2.0/UDP ms5.conf.example.com;branch=z9hG4bK343gf4
     Max-Forwards: 70
     From: <sip:3402934234@example.com>;tag=5393k2312
     To: Carol <sip:carol@chicago.example.com>;tag=32331
     Call-ID: d432fa84b4c76e66710
     CSeq: 78654 BYE
     Content-Length: 0
```

4.12 Deleting a conference

A conference created using the Conference Factory URI is automatically deleted when the creator of the conference leaves the conference by sending a BYE.

If the focus allows the conference policy to be manipulated, it is possible for the conference to continue after the creator departs if the policy is changed. However, the default conference policy for conferences created using the Conference Factory URI is that the conference is deleted when the creator departs.

Figure 12 shows this call flow in which the creator Alice departs causing the conference to be deleted. Note that the order of sending BYEs and final NOTIFYs is not important.

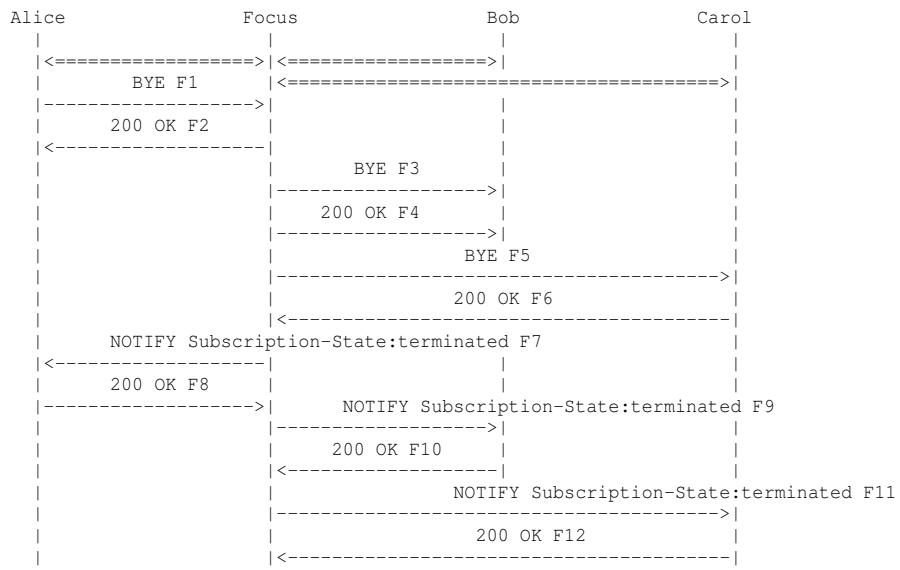


Figure 12. Deleting a Conference

4.13 Discovery of Conferencing Capabilities using OPTIONS

A UA MAY send an OPTIONS request to discover if an opaque URI is a conference URI (resolves to a focus). In addition, the reply to the OPTIONS request can also indicate support for various SIP call control extensions used in this document.

Note that the Allow, Accept, Allow-Events, and Supported header fields should be present in an INVITE from a focus or a 200 OK answer from the focus to an INVITE as a part of a normal dialog establishment process.

An example is shown in Figure 13 where Alice sends an OPTIONS to a URI which resolves to a focus.

Figure 13. Participant Queries Capabilities of URI which resolves to a Focus.

Following is an example message detail of message F2 in Figure 13. Based on the response, Alice's UA learns that the URI is a conference URI and that the responding UA is focus that supports a number of SIP call control extensions.

The response details are as follows:

```
F2   SIP/2.0 200 OK
     Via: SIP/2.0/UDP pc33.atlanta.example.com;branch=z9hG4bKhjhs8ass877
      ;received=192.0.2.4
     To: <sip:3402934234@example.com>;tag=93810874
     From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
     Call-ID: a84b4c76e66710
     CSeq: 63104 OPTIONS
     Contact: <sip:3402934234@example.com>;isfocus
     Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
      SUBSCRIBE, NOTIFY
     Allow-Events: refer, conference
     Accept: application/sdp, application/conference-info+xml,
      message/sipfrag
     Accept-Language: en
     Supported: replaces, join, gruu
     Content-Type: application/sdp
     Content-Length: ...

     v=0
     o=focus431 2890844563 2890842835 IN IP4 ms5.conf.example.com
     s=Example Subject
     i=Example Conference Hosted by Example.com
     u=http://conf.example.com/3402934234
     e=3402934234@conf-help.example.com
     p=+18882934234
     c=IN IP4 ms5.conf.example.com
     t=0 0
     m=audio 49170 RTP/AVP 0 1 3 5 7
     m=video 51372 RTP/AVP 31 32
```

Useful information from each of these headers is detailed in the next

sections.

Allow. The support of methods such as REFER, SUBSCRIBE, and NOTIFY indicate that the user agent supports call control and SIP Events.

Accept. The support of bodies such as message/sipfrag [14], application/conference-info+xml [5] also indicates support of call control and conferencing.

Allow-Events. The support of event packages such as refer [3], conference [5].

Supported. The support of extensions such as replaces, join, and gruu.

Contact. The presence of the "isfocus" feature parameter in the Contact header indicates that the URI is a conference URI and that the UA is a focus.

5. Appendix - Creating a Conference by a Conference-Unaware UA

This section discussed how a human user operating a conference-unaware UA can create and add participants to a conference. This method is described as an appendix since it is NOT RECOMMENDED. The scenarios involving creating a conference using ad-hoc or manual means are recommended over this scenario. This scenario is included, however, for completeness.

A user (human) would choose a conference URI according to system rules and insert it into the Request-URI of the INVITE. This same URI is echoed by a focus adhering to certain addressing conventions (discussed below) in the Contact header by the focus. Additional participants could be added by non-SIP means (publication of the chosen conference URI using web pages, email, IM, etc.). Alternatively, the conference-unaware UA could then add other participants to the conference using SIP call control by establishing a session with them, then transferring [17] them to the conference URI. Note that in this scenario only the user (human) is aware of the conferencing application, and the conference-unaware UA only need support RFC 3261 and optionally call transfer.

Making this work does impose certain addressing conventions on a system. As a service/implementation choice, a system could allow the creator of the conference to choose the user portion of the conference URI. However, this requires the URI format to be agreed upon between a user and the system.

For example, a service provider might reserve the domain

conf.example.com for all conference URIs. Any URI in the domain of conf.example.com would resolve to the focus. The focus could be configured to interpret an unknown user part in the conf.example.com domain as a request for a conference to be created with the conference URI as the Request-URI. For example, an INVITE sent with a Request-URI of sip:k32934208ds72@conf.example.com could be routed to the focus that would then create the conference. This conference URI should be registered by the newly created focus to become routable as a conference URI within the conf.example.com domain. The returned Contact would look as follows:

```
Contact: <sip:k32934208ds72@conf.example.com>;isfocus
```

Note, however, that this approach relies on conventions adopted between the user (human) and the focus. Also, the approach is not robust against collisions in the conference names. If a second user wishing to create a new conference happened to choose the same user part as an existing conference, the result would be that the second user would be added into the existing conference instead of creating a new one.

As a result, methods of conference creation in which the conference URI is an opaque URI generated by the focus are preferred.

An example call flow is shown in Figure 14. The participant Alice creates the conference URI (using some convention agreed to with the focus domain) and sends an INVITE to that URI which creates the focus. The focus creates the conference and returns the same conference URI in the 200 OK answer to the INVITE (which is ignored by the conference-unaware UA).

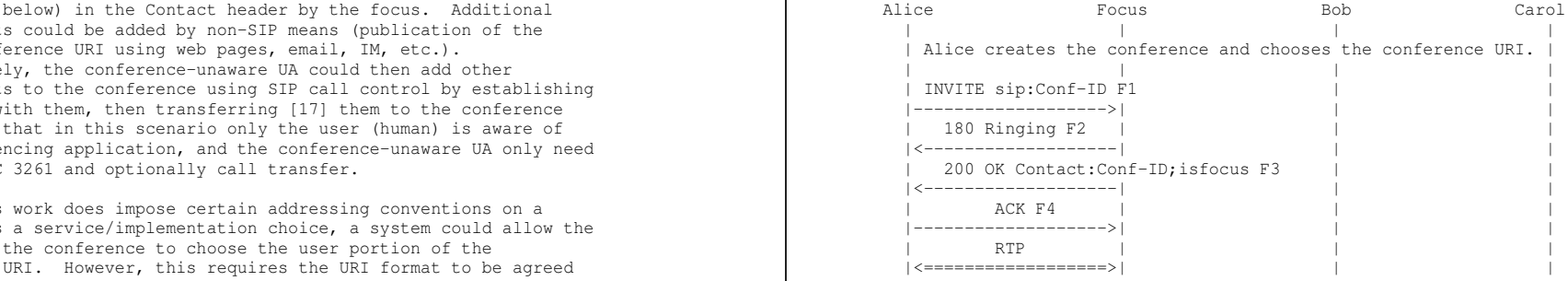


Figure 14. A Conferencing Unaware Participant Creates a Conference

6. IANA Considerations

This specification registers a single SIP option tag, conf.

6.1 SIP Option Tag

The required information for this registration, as specified in RFC 3261 [2], is:

Name: conf

Description: This option tag is used in a Supported header field of a request or response to indicate that the User Agent is conference-aware and supports the extensions and scenarios described in this specification.

7. Security Considerations

This document discusses call control for SIP conferencing. Both call control and conferencing have specific security requirements which will be summarized here. Conferences generally have authorization rules about who may or may not join a conference, what type of media may or may not be used, etc. This information is used by the focus to admit or deny participation in a conference. It is recommended that these types of authorization rules be used to provide security for a SIP conference. For this authorization information to be used, the focus needs to be able to authenticate potential participants. Normal SIP mechanisms including Digest authentication and certificates can be used. These conference specific security requirements are discussed further in the requirements and framework documents.

For call control security, a user agent must maintain local policy on who is permitted to perform call control operations, initiate REFERs, and replace dialogs. Normal SIP authentication mechanisms are also appropriate here. The specific authentication and authorization schemes are described in the multiparty call control framework document.

8. Contributors

We would like to thank Rohan Mahy, Jonathan Rosenberg, Roni Even, Petri Koskelainen, Brian Rosen, Paul Kyzivat, Eric Burger, and others in list discussions.

9. Changes since -03

- Added definition and IANA registration for conf option tag for UAs

- Added requirement and flow for deletion of conference

- Added scenario of participant requesting focus refer a participant to the conference

- Added scenario of moving a point to point call to a conference with a separate focus

10. Changes since -02

- Added reference and text about use of GRUUs.

- Updated for latest version of conference package.

- Clarified that conference package subscription should use a separate dialog from INVITE dialog.

11. Changes since -01

- Added messages details of selected INVITE, 200 OK, SUBSCRIBE, REFER, and NOTIFY messages.

12. Changes since -00

- Showed separation between conference factory application and focus by having the application redirect to the newly created focus in the ad-hoc creation scenario.

- Removed inclusion of "isfocus" parameter in Refer-To header field - this may be a useful extension to the REFER mechanism in the future, however.

- Updated reference from Caller Prefs document to the new Capabilities of User Agents document.

- Added scenario of participant changing user agents during a conference.

- Added requirement on focus to support Replaces header field.

- Added discussion about termination of dialog using BYE and subscription using SUBSCRIBE/NOTIFY to flows involving termination of session with the focus.

13. References

13.1 Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

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

[5] Rosenberg, J. and H. Schulzrinne, "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-04 (work in progress), May 2004.

[6] Mahy, R. and D. Petrie, "The Session Inititation Protocol (SIP) 'Join' Header", draft-ietf-sip-join-03 (work in progress), February 2004.

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

[8] Biggs, B., Dean, R. and R. Mahy, "The Session Inititation Protocol (SIP) 'Replaces' Header", draft-ietf-sip-replaces-05 (work in progress), February 2004.

[9] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-02 (work in progress), July 2004.

13.2 Informative References

[10] Levin, O. and R. Even, "High Level Requirements for Tightly Coupled SIP Conferencing", draft-ietf-sipping-conferencing-requirements-00 (work in progress), April 2003.

[11] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-02 (work in progress), June 2004.

[12] Mahy, R., "A Call Control and Multi-party usage framework for the Session Initiation Protocol (SIP)", draft-ietf-sipping-cc-framework-03 (work in progress), October 2003.

[13] Campbell, B. and R. Sparks, "Control of Service Context using SIP Request-URI", RFC 3087, April 2001.

[14] Sparks, R., "Internet Media Type message/sipfrag", RFC 3420, November 2002.

[15] Johnston, A., Donovan, S., Sparks, R., Cunningham, C. and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples", BCP 75, RFC 3665, December 2003.

[16] Johnston, A. and R. Sparks, "Session Initiation Protocol Service Examples", draft-ietf-sipping-service-examples-06 (work in progress), February 2004.

[17] Sparks, R. and A. Johnston, "Session Initiation Protocol Call Control - Transfer", draft-ietf-sipping-cc-transfer-02 (work in progress), February 2004.

[18] Rosenberg, J. and H. Schulzrinne, "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-dialog-package-04 (work in progress), February 2004.

Authors' Addresses

Alan Johnston
MCI
100 South 4th Street
St. Louis, MO 63102

EMail: alan.johnston@mci.com

Orit Levin
Microsoft
One Microsoft Way
Redmond, WA 75024

EMail: oritl@microsoft.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 procedures with respect to rights in RFC 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.

SIPPING
Internet-Draft
Expires: January 16, 2005

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia University
O. Levin, Ed.
Microsoft Corporation
July 18, 2004

A Session Initiation Protocol (SIP) Event Package for Conference State

draft-ietf-sipping-conference-package-05

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 16, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document defines a conference event package for the Session Initiation Protocol (SIP) Events framework, along with a data format used in notifications for this package. The conference package allows users to subscribe to a conference URI. Notifications are sent about changes in the membership of this conference and optionally about changes in the state of additional conference components.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [6] Events framework Events Framework [7] defines general mechanisms for subscribing to, and receiving notifications of, events within SIP networks. It introduces the notion of a package, which is a specific "instantiation" of the events framework for a well-defined set of events. Here, we define an event package for SIP conferences. This package provides the conference notification service as outlined in the SIP conferencing framework [14]. As described there, subscriptions to a conference URI are routed to the focus that is handling the conference. It acts as the notifier, and provides clients with updates on conference state.

The information provided by this package is comprised of conference identifier(s), conference participants (optionally with their statuses and media description), conference sidebars, conference policy URIs, etc.

2. Terminology

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 [2] and indicate requirement levels for compliant implementations.

3. Conference Event Package

The conference event package allows a user to subscribe to a conference. In SIP, conferences are represented by URIs. These URIs route to a SIP user agent, called a focus, that is responsible for ensuring that all users in the conference can communicate with each other, as described in Conferencing Framework [14]. The focus has sufficient information about the state of the conference to inform subscribers about it.

It is possible a participant in the conference may in fact be another focus. In order to provide a more complete participant list, the focus MAY subscribe to the conference package of the other focus to discover the participant list in the cascaded conference. This information can then be included in notifications by using of the "cascaded-focus" attribute as specified by this package.

This section provides the details for defining a SIP Events package, as specified by RFC 3265 [7].

3.1 Event Package Name

The name of this event package is "conference". This package name is carried in the Event and Allow-Events header, as defined in RFC 3265 [7].

3.2 SUBSCRIBE Bodies

A SUBSCRIBE for a conference package MAY contain a body. This body defines a filter to apply to the subscription. Filter documents are not specified in this document, and at the time of writing, are expected to be the subject of future standardization activity.

A SUBSCRIBE for a conference package MAY be sent without a body. This implies the default subscription filtering policy. The default policy is:

- Notifications are generated every time there is any change in the state of the conference.
- Notifications do not normally contain full state; rather, they only indicate the state that has changed. The exception is a NOTIFY sent in response to a SUBSCRIBE. These NOTIFYs contain the full state of the information requested by the subscriber.

3.3 Subscription Duration

The default expiration time for a subscription to a conference is one hour. Once the conference ends, all subscriptions to that particular conference are terminated, with a reason of "noresource" RFC 3265

[7].

3.4 NOTIFY Bodies

As described in RFC 3265 [7], the NOTIFY message will contain bodies that describe the state of the subscribed resource. This body is in a format listed in the Accept header field of the SUBSCRIBE, or a package-specific default if the Accept header field was omitted from the SUBSCRIBE.

In this event package, the body of the notification contains a conference information document. This document describes the state of a conference. All subscribers and notifiers MUST support the "application/conference-info+xml" data format described in Section 4. The subscribe request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/ conference-info+xml". If the header field is present, it MUST include "application/conference-info+xml", and MAY include any other types capable of representing dialog state.

Of course, the notifications generated by the server MUST be in one of the formats specified in the Accept header field in the SUBSCRIBE request.

3.5 Notifier Processing of SUBSCRIBE Requests

The conference information contains very sensitive information. Therefore, all subscriptions SHOULD be authenticated and then authorized before approval. Authorization policy is at the discretion of the administrator, as always. However, a few recommendations can be made.

It is RECOMMENDED that all users in the conference be allowed to subscribe to the conference.

3.6 Notifier Generation of NOTIFY Requests

Notifications SHOULD be generated for the conference whenever there is a change in the state in any of the information delivered to the subscriber.

The changes generally occur when a new participant joins (i.e. gets "connected" to) or a participant leaves (i.e. gets "disconnected" from) the conference.

Subject to a local focus policy, additional changes in participant's status, changes in its media types, and other optional media attributes MAY be reported by the focus.

Changes in sidebar rosters SHOULD be reported by the focus to their participants and MAY be reported to others, subject to local policy.

Changes in conference identifiers and policy URIs SHOULD be reported by the focus to the conference participants.

3.7 Subscriber Processing of NOTIFY Requests

The SIP Events framework expects packages to specify how a subscriber processes NOTIFY requests in any package specific ways, and in particular, how it uses the NOTIFY requests to construct a coherent view of the state of the subscribed resource.

Typically, the NOTIFY for the conference package will only contain information about those users whose state in the conference has changed. To construct a coherent view of the total state of all users, a subscriber to the conference package will need to combine NOTIFYs received over time.

Notifications within this package can convey partial information; that is, they can indicate information about a subset of the state associated with the subscription. This means that an explicit algorithm needs to be defined in order to construct coherent and consistent state. The details of this mechanism are specific to the particular document type. See Section 4.2 for information on constructing coherent information from an application/ conference-info+xml document.

3.8 Handling of Forked Requests

By their nature, the conferences supported by this package are centralized. Therefore, SUBSCRIBE requests for a conference should not generally fork. Users of this package MUST NOT install more than a single subscription as a result of a single SUBSCRIBE request.

3.9 Rate of Notifications

For reasons of congestion control, it is important that the rate of notifications not become excessive. As a result, it is RECOMMENDED that the server not generate notifications for a single subscriber at a rate faster than once every 5 seconds.

3.10 State Agents

Conference state is ideally maintained in the element in which the conference resides. Therefore, the elements that maintain the conference are the ones best suited to handle subscriptions to it. Therefore, the usage of state agents is NOT RECOMMENDED for this

package.

4. Conference Data Format

Conference information is an XML document that MUST be well-formed and SHOULD be valid. Dialog information documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying dialog information documents and document fragments. The namespace URI for elements defined by this specification is a URN [3], using the namespace identifier 'ietf' defined by [4] and extended by [1]. This URN is:

urn:ietf:params:xml:ns:conference-info

A conference information document begins with the root element tag "conference-info".

4.1 Conference Information

Conference information begins with the top level element "conference-info". This element has three mandatory attributes:

version: This mandatory attribute allows the recipient of conference information documents to properly order them. Versions start at 0 and increment by one for each new document sent to a subscriber. Versions are scoped within a subscription. Versions MUST be represented using a 32 bit integer.

state: This mandatory attribute indicates whether the document contains the full conference information, or whether it contains only the information that has changed since the previous document (partial).

entity: This mandatory attribute contains the conference URI that identifies the conference being described in the document.

The "conference-info" element has zero or more "user" sub-elements which contain information on the users in the conference. This is followed by zero or more "sidebar" sub-elements which contain information on the sidebars in the conference. This is followed by zero or more "conf-uri" sub-elements which contain information on additional URIs that the conference can be accessed by. This is followed by zero or more "policy-uri" sub-elements which contain information on additional URIs that the conference policies can be accessed by. This is followed by "recording" and "streaming" elements describing recording and streaming statuses of the conference.

4.1.1 User Element

4.1.1.1 User Attributes

The user element has one mandatory attribute, "uri" that indicates the URI for the user in the conference. This is a logical identifier, which corresponds to the authenticated identity of the participant. The "uri" attribute MUST be unique in the user element list because it is used as the key in partial notifications about users' state.

If a conference participant has more than a single signaling dialog associated with the conference, the conference focus MAY present the user's aggregated information (e.g. the statuses) and display all its media streams under a single user element.

Note, that the optional element "instance" of "media" (see below) MAY be used in this case to specify the actual signaling dialog for each media stream.

An anonymous participant in a conference SHOULD be represented by an anonymous URI generated by the focus. For multiple anonymous participants, the focus must ensure that each anonymous URI is unique. The guidelines for generating anonymous URIs in RFC 3323 [8] should be followed. For example,

"Anonymous1" <sip:anonymous1@anonymous.invalid>

could be used for a participant requesting privacy.

The optional attribute "display-name" contains a display name for the user. The standard "xml:lang" language attribute can also be present to indicate the language of the display-name.

The optional attribute "cascaded-focus" contains a conference URI (different from the main conference URI) for users that are connected to the main conference as a result of focus cascading. In accordance with the SIP conferencing framework [14], this package allows for representation of peer-to-peer (i.e. "flat") focus cascading only. The actual cascading graph can not be deduced from the information provided in the package alone. Advanced applications can construct the graph by subscribing to both this package and the Dialog Package [15] of the cascaded foci and correlating the relevant information.

If the main conference "state" is "full", the state of its user(s) MUST "full". If the main conference "state" is "partial", the state of its user(s) MAY be either "partial" or "full".

4.1.1.2 User Status Elements

Three optional status elements are defined: status, joining-mode, and disconnection-reason.

- "status": provides information about user's current level of participation in the conference.
- "joining-mode": if present, provides information about the way the user joined the conference.
- "disconnection-reason": if present, provides information about the way the user left the conference.

The following statuses are defined for the "status" element:

connected: The user is a participant in the conference. Depending on the media policies, he/she can send and receive media to and from other participants.

disconnected: The user is not a participant in the conference and no active dialog exists between the user and the focus.

on-hold: Active SIP dialog exists between a user and a focus, but user is "on-hold" for this conference, i.e. neither he/she is "hearing" the conference mix, nor is his/her media being mixed in the conference. As an example, the user has asked to join the conference using SIP, but his/her participation is pending based on moderator approval. In the meantime he/she is hearing music-on-hold or some other kind of related content.

muted-via-focus: Active SIP dialog exists between a user and a focus and the user can "listen" to the conference, but user's media is not being mixed into the conference. Note that sometimes a subset of user media streams can be muted by focus (such as poor quality video) while others (such as voice or IM) can still be active. In this case, it is RECOMMENDED that the "aggregated" user connectivity "status" reflects the status of the mostly active media.

blocked: User is denied from ever participating in this conference.

pending: User is not yet in the session, but it is anticipated that he/she will join in the near future.

calling: User is being called by the focus.

ringing: An PSTN ALERTING or SIP 180 Ringing was returned for the outbound call, user is being alerted.

dialing-in: User is dialing into the conference, not yet in the roster (probably being authenticated).

disconnecting: Focus is in the process of disconnecting user (either DISCONNECT or BYE was sent to the user's device).

removed: This status is used to remove the user from the roster using partial notifications mechanism.

Note that the defined transient states (e.g., calling, ringing, etc.) could generate a lot of notifications. Implementations MAY choose not to generate notifications on these to all participants if it will

generate too much traffic.

The following statuses are defined for the "joining-mode" element:
dialed-in: The user dialed into the conference, i.e. sent INVITE to the focus, which resulted in successful dialog establishment.
dialed-out: The focus has brought the user into the conference by sending a successful INVITE to the user.
focus-owner: The user is the focus for this conference. This status is used only when a participant UA acts as a conference focus.

The following statuses are defined for the disconnection-reason element:
departed: The user sent a BYE, thus leaving the conference.
booted: The user was sent a BYE by the focus, booting him/her out of the conference. Alternatively, the user tried to dial into to conference without success because was rejected by the focus according to local policy decisions.
failed: The server tried to bring the user into the conference, but its attempt to contact the specific user resulted in a non-200 class final response. Alternatively, the user tried to dial into the conference without success due to technical reasons.

4.1.1.3 Media Information

Each user has zero or more "media" sub-elements.

Each "media" element indicates the media that the user is currently connected to. Here, "connected to" implies that a user has a media line in his/her SDP [12] document(s). With this definition, a user is connected to a media stream even if he/she is not sending any media.

4.1.1.3.1 Media Attributes

The "media" element has a mandatory "media-type" attribute which identifies the media type (e.g. audio, video, message and application) and MUST have one of the values registered for "media" of SDP [12].

The optional "id" attribute serves as a unique reference to a "media" element within the "user" element. It MUST be included for each "media" element for all notifications if the focus uses "partial" user notifications for this conference. Otherwise, the "id" attribute MAY be omitted.

If the user "state" is "full", the state of its "media" element(s) MUST be "full". If the user "state" is "partial", the state of its "media" element(s) MAY be either "partial" or "full".

4.1.1.3.2 Media Elements

The "media" element has also an optional "proto" sub-element, which MUST has the value registered for "proto" of SDP [12].

An optional "ssrc" sub-element, if present, carries the value of SSRC (defined in RTP/RTCP [10]) as generated by the user for the stream it sends.

When an RTP mixer generates a CSRC list according to RTP/RTCP [10], it inserts a list of the SSRC identifiers of the sources that contributed to the generation of a particular packet into the RTP header of that packet. "An example application is audio conferencing where a mixer indicates all the talkers whose speech was combined to produce the outgoing packet, allowing the receiver to indicate the current talker, even though all the audio packets contain the same SSRC identifier (that of the mixer)."

An optional "info" sub-element, if present, carries a human readable description for this stream populated by the focus. The value of this element corresponds to the information media attribute "i" in SDP [12].

An optional "label" sub-element, if present, carries a unique identifier for this stream among all streams in the conference and is assigned by the focus. The value of this element corresponds to the "label" media attribute in SDP [12] and defined in [18].

An optional "instance" sub-element, if present, carries a URI, which MUST uniquely identify the signaling dialog being used for establishing of this media stream. In SIP, for example, values of Contact URI or GRUU [17] can be used for this purpose. It is RECOMMENDED to include the "instance" information for every user that has more than a single dialog associated with the conference. This element SHOULD NOT be included for an anonymous participant.

An optional "status" sub-element, if present, is used to remove "media" elements during partial notifications.

Optional "snd-status" and "rcv-status" sub-elements, if present, describe the status of media streams in each direction.

4.1.1.4 User Role

The optional "role" element conveys the role of the user in the conference, e.g. participant, presenter, panelist, host, etc. User's role MAY change dynamically in the course of the conference. Also, a user MAY have more than a single role in one time.

This document does not define fixed values for the "role" element, instead it is expected that conferencing applications will define custom-fit roles by templates.

4.1.2 Sidebar Element

The sidebar element is of the general "conference-type" and MAY use all the attributes and elements defined by it. Typically, only the "entity", which uniquely identifies the sidebar, and the "user" elements will be useful to present to the majority of the participants in the conference.

The "conference-type" mandatory attributes MUST be included for each sidebar.

The value of the "version" attribute is meaningless for "sidebar" elements and MUST be ignored because it is always overruled by the "version" attribute of the main "conference-info".

If the main conference "state" is "full", the state of its sidebar(s) MUST be "full". If the main conference "state" is "partial", the state of its sidebar(s) MAY be either "full" or "partial".

The "entity" URI attribute MUST be unique among the sidebar identifiers of the same conference. Attribute "entity" is used as the key for "sidebar" elements in partial notifications for "conference-info".

4.1.3 Additional Conference Identifiers

In addition to the Conference URI present in the "entity" attribute, a conference MAY have additional URIs of various types. Connecting to these URIs will result in joining to the same conference.

4.1.4 Policy URIs

A policy URI specifies where and how a certain policy pertaining to the conference can be accessed. The actual policy name and usage is deduced from the URI schema name.

An example for the "policy-uri" usage is inclusion of the URI of the CPCP [16]. A subscriber to the Conference package can use the Policy URI to access and modify the conference policy.

4.1.5 Recording

In many cases, legal regulations require conference providers to announce to the participants that a specific conference is being

recorded.

In addition to the recording "status" information, the "recording" element MAY include the URIs specifying the location and the format of the recorded data. Typically, the recorded data becomes available after the conference ends. Multiple URIs can be provided, for example, specifying different content types. For Web-Page embedded media, a plain HTTP URI MAY be provided.

4.1.6 Streaming

The "streaming" element, if present, specifies whether the conference output is being streamed (to general public, for example), in what streaming format, and at what (e.g. multicast) addresses it can be listened at. RTSP [11] is an example of such streaming protocol.

4.2 Constructing Coherent State

The conference information is described by a hierarchal XML structure with the root element "conference-info". The root element is the only element in the schema that carries meaningful version number for all the elements in the document. The whole conference information is associated with this version number.

The version number MUST be initialized with the value of the "version" attribute from the "conference-info" element in the first document received. Each time a new document is received, the value of the local version number, and the "version" attribute in the new document, are compared. If the value in the new document is one higher than the local version number, the local version number is increased by one, and the document is processed. If the value in the document is more than one higher than the local version number, the local version number is set to the value in the new document, the document is processed, and the subscriber SHOULD generate a refresh request to trigger a full state notification. If the value in the document is less than the local version, the document is discarded without processing.

Further processing of the conference information document depends on whether it contains full or partial state. If it contains full state, indicated by the value of the "state" attribute in the "conference-info" element, the whole local content is flushed and repopulated from the document.

If the document contains partial state, as indicated by the value of the "state" attribute in the "conference-info" element, the document is used to update the local content as described below.

All sub-elements in the "conference-info" hierarchal XML structure can be classified in two groups: those that carry relatively small amount of data and those that can potentially carry a lot of data. During partial notifications, the light elements are updated as atomic pieces of data. On the other hand, elements that can carry a substantial amount of data have the general "state" attribute attached to them. That is in order to support partial notifications for their content.

A "state" attribute of a child element in the document MUST adhere to its parent "state". It means that if the parent's "state" is "full", the state of its children MUST be "full". If the parent's "state" is "partial", the state of its children MAY be either "partial" or "full".

For elements with optional "state" attribute, if the attribute is not included for an element, it means that the element's state is "full".

For a parent element with "state", its sub-elements with possible multiple appearances under the parent have keys that uniquely identify each element among others in the same list.

4.2.1 The Algorithm

The conference package subscriber locally maintains a local element for each element in the schema and a table for each "element with key(s)" in the schema. The tables are indexed by the key(s) defined in schema for the element.

Starting from outer elements in the received document,

1. If the parent element contains full state, the element is replaced with the new information as a whole.

2. Otherwise, if the parent element contains partial state,

2.1 For elements with keys, the subscriber compares the keys received in the update with the keys in the local tables.

2.1.1 If a key doesn't exist in the local table, a row is added, and its content is set to the element information from the update.

2.1.2 Otherwise, if a key of the same value does exist, for each sub-element in the row the algorithm is applied from step 2.2.

2.2 For each atomic element received in the schema, the element is replaced with the new information as a whole. Also, for each non-atomic element received in the schema with either no "state"

attribute included or the state attribute is set to "full", the element is replaced with the new information as a whole.

2.2.1 If the updated or created element carries the "removed" status, that element SHOULD be removed from the local content. If the element is updated or created, such that it is empty, that element MAY be removed from the local content at any time.

2.3 For each non-atomic element with the state attribute set to "partial", the algorithm is applied recursively starting from step 2.

4.3 Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:conference-info" xmlns:tns="urn:
<!--
 This import brings in the XML language attribute xml:lang
  -->
  <xs:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http:

<xs:element name="conference-info" type="tns:conference-type"/>

<xs:simpleType name="state-type">
 <xs:restriction base="xs:string">
       <xs:enumeration value="full" />
       <xs:enumeration value="partial" />
 </xs:restriction>
</xs:simpleType>

<xs:complexType name="conference-type">

<xs:sequence>
  <xs:element name="user" type="user-type" minOccurs="0" maxOccurs="unbounded" />
  <xs:element name="sidebar" type="conference-type" minOccurs="0" maxOccurs="unboun
  <xs:element name="conf-ids" type="conf-ids-type" minOccurs="0" maxOccurs="1" />
  <xs:element name="policy-ids" type="policy-ids-type" minOccurs="0" maxOccurs="1"
  <xs:element name="recording" type="recording-type" minOccurs="0" maxOccurs="1" />
  <xs:element name="streaming" type="streaming-type" minOccurs="0" maxOccurs="1" />
  <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>

  <xs:attribute name="version" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="state"   type="tns:state-type"        use="required"/>
  <xs:attribute name="entity"  type="xs:anyURI"             use="required"/>
  <xs:anyAttribute />
```

```
   </xs:complexType>

   <xs:complexType name="conf-ids-type">
    <xs:sequence>
     <xs:element name="conf-uri" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>

    <xs:anyAttribute />
   </xs:complexType>

   <xs:complexType name="policy-ids-type">
    <xs:sequence>
     <xs:element name="policy-uri" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>

    <xs:anyAttribute />
   </xs:complexType>

   <xs:complexType name="recording-type">
    <xs:sequence>
     <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded" />
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>

    <xs:attribute name="status" type=="stream-status-type" use="required"/>
    <xs:anyAttribute />
   </xs:complexType>

   <xs:complexType name="streaming-type">
    <xs:sequence>
     <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded" />
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>

    <xs:attribute name="status" type="stream-status-type" use="required"/>
    <xs:anyAttribute />
   </xs:complexType>

   <xs:complexType name="user-type">
    <xs:sequence>
     <xs:element name="status" type="tns:user-status-type" minOccurs="0"/>
     <xs:element name="joining-mode" type="tns:user-joining-mode-type" minOccurs="0"/>
     <xs:element name="disconnection-reason" type="tns:user-disconnection-reason-type"
     <xs:element name="media" type="tns:media-type" minOccurs="0" maxOccurs="unbounded
     <xs:element name="role" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```



```
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>

    </xs:sequence>

    <xs:attribute name="uri"          type="xs:anyURI"   use="required"/>
    <xs:attribute name="display-name" type="xs:string"   use="optional"/>
    <xs:attribute ref="xml:lang"                         use="optional"/>
    <xs:attribute name="cascaded-focus" type="xs:anyURI" use="optional"/>
    <xs:attribute name="state" type="tns:state-type"     use="optional"/>
    <xs:anyAttribute />

   </xs:complexType>

   <xs:simpleType name="user-status-type">
    <xs:restriction base="xs:string">
     <xs:enumeration value="connected"/>
     <xs:enumeration value="disconnected"/>
     <xs:enumeration value="on-hold"/>
     <xs:enumeration value="muted-via-focus"/>
     <xs:enumeration value="blocked"/>
     <xs:enumeration value="pending"/>
     <xs:enumeration value="calling"/>
     <xs:enumeration value="ringing"/>
     <xs:enumeration value="dialing-in"/>
     <xs:enumeration value="disconnecting"/>
     <xs:enumeration value="removed"/>
    </xs:restriction>
   </xs:simpleType>

   <xs:simpleType name="user-joining-mode-type">
    <xs:restriction base="xs:string">
     <xs:enumeration value="dialed-in" />
     <xs:enumeration value="dialed-out" />
     <xs:enumeration value="focus-owner" />
    </xs:restriction>
   </xs:simpleType>

   <xs:simpleType name="user-disconnection-reason-type">
    <xs:restriction base="xs:string">
     <xs:enumeration value="departed" />
     <xs:enumeration value="booted" />
     <xs:enumeration value="failed" />
    </xs:restriction>
   </xs:simpleType>

   <xs:complexType name="media-type">
    <xs:sequence>
     <xs:element name="proto"      type="xs:string"            minOccurs="0"/>
```

```
     <xs:element name="ssrc"       type="xs:nonNegativeInteger" minOccurs="0"/>
     <xs:element name="info"       type="xs:string"             minOccurs="0"/>
     <xs:element name="label"      type="xs:string"             minOccurs="0"/>
     <xs:element name="instance"   type="xs:anyURI"             minOccurs="0"/>
     <xs:element name="status"     type="tns:media-status-type" minOccurs="0"/>
     <xs:element name="snd-status" type="tns:stream-status-type" minOccurs="0"/>
     <xs:element name="rcv-status" type="tns:stream-status-type" minOccurs="0"/>
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>

     <xs:attribute name="media"  type="xs:string"            use="required"/>
     <xs:attribute name="id"     type="nonNegativeInteger"   use="optional"/>
     <xs:attribute name="state"  type="tns:state-type"       use="optional"/>
     <xs:anyAttribute />

   </xs:complexType>

   <xs:simpleType name="media-status-type">
    <xs:restriction base="xs:string">
     <xs:enumeration value="removed" />
    </xs:restriction>
   </xs:simpleType>

   <xs:simpleType name="stream-status-type">
    <xs:restriction base="xs:string">
     <xs:enumeration value="on"/>
     <xs:enumeration value="off"/>
     <xs:enumeration value="muted" />
    </xs:restriction>
   </xs:simpleType>

   </xs:schema>
```

4.4 Example

The following is an example conference information document:


```
<?xml version="1.0" encoding="utf-8" ?>
<conference-info version="0" state="full" entity="sip:conf233@example.com">
      <user uri="sip:bob@example.com" display-name="Bob Jones">
            <status>connected</status>
            <joining-mode>dialed-in</joining-mode>
            <media media="audio">
                  <proto>RTP/AVP</proto>
                  <ssrc>583398</ssrc>
            </media>
      </user>
      <user uri="sip:barbara@example.com" display-name="Barbara Jones">
            <status>on-hold</status>
      </user>
      <user uri="sip:bill@example.com" display-name="Bill Minelli">
            <status>on-hold</status>
      </user>

      <sidebar version="0" state="full" entity="sip:conf233.1@example.com">
            <user uri="sip:barbara@example.com" />
            <user uri="sip:bill@example.com" />
      </sidebar>

      <conf-ids>
            <conf-uri>tel:+18005671234</conf-uri>
            <conf-uri>h323:conf545@example.com</conf-uri>
      </conf-ids>

      <recording status="on">
            <uri>http://quicktime.streaming.com/54634/recording.mov</uri>
            <uri>http://real.streaming.com/54634/recording.ram</uri>
            <uri>http://windowsmedia.streaming.com/54634/recording.wmv</uri>
            <uri>http://www.streaming.com/54634/recording.html</uri>
      </recording>

</conference-info>
```

This conference currently has three users, two of which are in a sidebar conversation. The conference is being recorded. There are additional means to join the conference either by phone using tel URI [14] or by H.323 protocol using H.323 URL [13].

5. Security Considerations

Subscriptions to conference state can reveal very sensitive information. For this reason, the document recommends authentication and authorization, and provides guidelines on sensible authorization policies.

Since the data in notifications is sensitive as well, end-to-end SIP encryption mechanisms using S/MIME SHOULD be used to protect it.

Since a focus provides participants identity information using this event package, participant privacy needs to be taken into account. A focus MUST support requests by participants for privacy. Privacy can be indicated by the conference policy - for every participant or select participants. It can also be indicated in the session signaling. In SIP this can be done using the Privacy header field described in RFC 3323 [8]. For a participant requesting privacy, no identity information SHOULD be revealed by the focus such as a URI (e.g. the Address of Record, Contact, or GRUU). For these cases, the anonymous URI generation method outlined in section "User Element" of this document MUST be followed.

6. IANA Considerations

This document registers a SIP event package, a new MIME type, application/conference-info+xml, a new XML namespace, and a new XML schema.

6.1 conference Event Package Registration

This specification registers an event package, based on the registration procedures defined in RFC 3265 [7]. The following is the information required for such a registration:
Package Name: conference
Package or Template-Package: This is a package.
Published Document: RFC XXXX (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification).
Person to Contact: Jonathan Rosenberg, jdrosen@jdrosen.net.

6.2 application/conference-info+xml MIME Registration

MIME media type name: application
MIME subtype name: conference-info+xml
Mandatory parameters: none
Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [5].
Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [5].
Security considerations: See Section 10 of RFC 3023 [5] and Section 5 of this specification.
Interoperability considerations: none.
Published specification: This document.
Applications which use this media type: This document type has been used to support SIP conferencing applications.
Additional Information:
Magic Number: None
File Extension: .cif or .xml
Macintosh file type code: "TEXT"
Personal and email address for further information: Jonathan Rosenberg, <jdrosen@jdrosen.net>
Intended usage: COMMON
Author/Change controller: The IETF.

6.3 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:conference-info

This section registers a new XML namespace, as per the guidelines in [1].

URI: The URI for this namespace is
urn:ietf:params:xml:ns:conference-info.
Registrant Contact: IETF, SIPPING working group, <sipping@ietf.org>,
Jonathan Rosenberg <jdrosen@jdrosen.net>.
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>Conference Information Namespace</title>
</head
<body>
  <h1>Namespace for Conference Information</h1>
  <h2>urn:ietf:params:xml:ns:conference-info</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

6.4 XML Schema Registration

This specification registers a schema, as per the guidelines in in [1].

URI: please assign.
Registrant Contact: IETF, SIPPING Working Group
(sipping@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).
XML: The XML can be found as the sole content of Section 4.3.

7. Acknowledgements

The authors would like to thank Dan Petrie, Sean Olson, Alan Johnston, and Rohan Mahy for their comments and inputs.

8. Changes History

8.1 Changes since -04

- o
- o "Sidebar-type" has been removed. "Sidebar" conference element is defined using the general "conference-type".
- o "Recording" conference attribute has been replaced with "recording" and "streaming" elements within "conference-type". New "recording-type" and "streaming-type" have been introduced.
- o Attribute "state" has been added to "user-type".
- o Element "media-stream" within "user-type" has been renamed to "media".
- o Element "role" within "user-type" has been introduced.
- o The following statuses have been added to "user-status-type": blocked, pending, calling, ringing, dialing-in, disconnecting, removed.
- o User status "muted-by-focus" has been renamed to "muted-via-focus".
- o Attributes "id" and "state" have been added to "media-type".
- o Elements "status", "snd-status" and "rcv-status" have been added to "media-type".
- o Element "dialog-id" has been renamed to "instance".
- o "Constructing Coherent State" section has been updated to include user and media partial notifications.

8.2 Changes since -03

- o "Constructing Coherent State" section has been updated.
- o In order to support partial notifications, two placeholders "conference-ids" and "policy-ids" (for "conf-uri" and "policy-uri" elements, correspondingly) are created.
- o Discussion and security considerations regarding anonymous participation have been added.
- o Optional elements "dialog-uri", "info" and "label" per media stream are added.

8.3 Changes since -02

- o State "muted-by-focus" is added to user's status.
- o Optional conference attribute "recording" is added.
- o Policy URI placeholder (i.e. element "policy-uri") is created.
- o Example's syntax is corrected.
- o Optional attribute "cascaded-focus" URI per user is added.
- o Optional additional conference identifiers (i.e. element "conf-uri") are added.
- o In order to cover all possible cases, participant's status is expressed using three optional statuses: "status", "joining-mode" and "disconnection-reason". That is instead of "activity-status", "history-status" and "is-on-dial-out-list".

8.4 Changes since -01

- o Package parameters are removed. Decision about performing "recursive" membership algorithm is perceived as a focus local policy.
- o General information (i.e. pointers to additional available services) is removed. The defined XML schema can be extended in future to include those when XCON work matures.
- o Dialog information is removed. It can be obtained by direct subscription to a dialog package of a participant.
- o Media stream information is aligned with SDP definitions (media and proto) and SSRC attribute is added.
- o Participant's status is expressed using two optional statuses: "activity" and "history". Optional "is-on-a-dial-out-list" indication is added.
- o Normative references to XCON work are removed.
- o Optional sidebar rosters are added.

9. References

9.1 Normative References

[1] Mealling, M., "The IETF XML Registry", draft-mealling-iana-xmlns-registry-05 (work in progress), June 2003.

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

[3] Moats, R., "URN Syntax", RFC 2141, May 1997.

[4] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[5] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[6] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

[8] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November 2002.

[9] Camarillo, G., Eriksson, G., Holler, J. and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.

[10] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

9.2 Informative References

[11] Schulzrinne, H., Rao, A. and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[12] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[13] Levin, O., "H.323 Uniform Resource Locator (URL) Scheme Registration", RFC 3508, April 2003.

[14] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-02 (work in progress), June 2004.

[15] Rosenberg, J. and H. Schulzrinne, "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-dialog-package-04 (work in progress), February 2004.

[16] Koskelainen, P. and H. Khartabil, "Requirements for Conference Policy Control Protocol", draft-ietf-xcon-cpcp-reqs-03 (work in progress), April 2004.

[17] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-02 (work in progress), July 2004.

[18] Levin, O. and G. Camarillo, "The SDP (Session Description Protocol) Label Attribute", draft-levin-mmusic-sdp-media-label-00 (work in progress), July 2004.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027
US

EMail: schulzrinne@cs.columbia.edu
URI: http://www.cs.columbia.edu/~hgs

Orit Levin (editor)
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

EMail: oritl@microsoft.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 procedures with respect to rights in RFC 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.

A Framework for Session Initiation Protocol User Agent Profile Delivery

draft-ietf-sipping-config-framework-04.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 17, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document defines the application of a set of protocols for providing profile data to SIP user agents. The objective is to define a means for automatically providing profile data a user agent needs to be functional without user or administrative intervention. The framework for discovery, delivery, notification and updates of user agent profile data is defined here. As part of this framework a new SIP event package is defined here for the notification of profile changes. This framework is also intended to ease ongoing administration and upgrading of large scale deployments of SIP user agents. The contents and format of the profile data to be defined is outside the scope of this document.

Table of Contents

1. Motivation

Today all SIP user agent implementers use proprietary means of delivering user or device profiles to the user agent. The profile delivery framework defined in this document is intended to enable a first phase migration to a standard means of providing profiles to SIP user agents. It is expected that UA implementers will be able to use this framework as a means of delivering their existing proprietary user and device data profiles (i.e. using their existing proprietary binary or text formats). This in itself is a tremendous advantage in that a SIP environment can use a single profile delivery server for profile data to user agents from multiple implementers. Follow-on standardization activities can:

1. define a standard profile content format framework (e.g. XML with namespaces [W3C.REC-xml-names11-20040204] or name-value pairs [RFC0822]).
2. specify the content (i.e. name the profile data parameters, xml schema, name spaces) of the data profiles.

One of the objectives of the framework described in this document is to provide a start up experience similar to that of users of an analog telephone. When you plug in an analog telephone it just works (assuming the line is live and the switch has been provisioned). There is no end user configuration required to make analog phone work, at least in a basic sense. So the objective here is to be able to take a new SIP user agent out of the box, plug it in or install the software and have it get its profiles without human intervention other than security measures. This is necessary for cost effective deployment of large numbers of user agents.

Another objective is to provide a scalable means for ongoing administration of profiles. Administrators and users are likely to want to make changes to user and device profiles.

Additional requirements for the framework defined in this document are described in: [I-D.ietf-sipping-ua-prof-framewk-reqs], [I-D.sinnreich-sipdev-req]

2. Introduction

2.1 Requirements Terminology

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in RFC 2119[RFC2119].

2.2 Profile Delivery Framework Terminology

profile - data set specific to a user or device.
device - SIP user agent, either software or hardware appliance.
profile content server - The server that provides the content of the profiles using the protocol specified by the URL scheme.
notifier - The SIP user agent server which processes SUBSCRIBE requests for events and sends NOTIFY requests with profile data or URI(s) point to the data.
profile delivery server - The logical collection of the SIP notifier and the server which provides the contents of the profile URI(s).

2.3 Overview

The profile life cycle can be described by five functional steps. These steps are not necessarily discrete. However it is useful to describe these steps as logically distinct. These steps are named as follows:

Discovery - discover a profile delivery server
Enrollment - enroll with the profile delivery server
Profile Retrieval - retrieve profile data
Profile Change Notification - receive notification of profile changes
Profile Change Upload - upload profile data changes back to the profile delivery server

Discovery is the process by which a UA finds the address and port at which it enrolls with the profile delivery server. As there is no single discovery mechanism which will work in all network environments, a number of discovery mechanisms are defined with a prescribed order in which the UA tries them until one succeeds.

Enrollment is the process by which a UA makes itself known to the profile delivery server. In enrolling the UA provides identity information, name requested profile type(s) and supported protocols for profile retrieval. It also subscribes to a mechanism for notification of profile changes. As a result of enrollment, the UA receives the data or the URI for each of the profiles that the profile delivery server is able to provide. Each profile type (set) requires a separate enrollment or SUBSCRIBE session.

Profile Retrieval is the process of retrieving the content for each of the profiles the UA requested.

Profile Change Notification is the process by which the profile delivery server notifies the UA that the content of one or more of the profiles has changed. If the content is provided indirectly the UA SHOULD retrieve the profile from the specified URI upon receipt of

the change notification.

Profile Upload is the process by which a UA or other entity (e.g. OSS, corporate directory or configuration management server) pushes a change to the profile data back up to the profile delivery server.

This framework defines a new SIP event package [RFC3265] to solve enrollment and profile change notification steps. This event packet defines everything but the mandatory content type. This make this event package abstract until the content type is bound. The profile content type(s) will be defined outside the scope of this document. It is he author's belief that it would be a huge accomplishment if all SIP user agent used this framework for delivering their existing proprietary profiles. Even though this does not accomplish interoperability of profiles, it is a big first step in easing the administration of SIP user agents. The definition of standard profiles and data set (see [I-D.petrie-sipping-profile-datasets]) will enable interoperability as a subsequent step.

The question arises as to why SIP should be used for the profile delivery framework. In this document SIP is used for only a small portion of the framework. Other existing protocols are more appropriate for transport of the profile contents (to and from the user agent) and are suggested in this document. The discovery step is simply a specified order and application of existing protocols. SIP is only needed for the enrollment and change notification functionality of the profile delivery framework. In many SIP environments (e.g. carrier/subscriber and multi-site enterprise) firewall, NAT and IP addressing issues make it difficult to get messages between the profile delivery server and the user agent requiring the profiles.

With SIP the users and devices already are assigned globally routable addresses. In addition the firewall and NAT problems are already presumably solved in the environments in which SIP user agents are to be used. Therefore SIP is the best solution for allowing the user agent to enroll with the profile delivery server which may require traversal of multiple firewalls and NATs. For the same reason the notification of profile changes is best solved by SIP.

The content delivery server may be either in the public network or accessible through a DMZ. The user agents requiring profiles may be behind firewalls and NATs and many protocols, such as HTTP, may be used for profile content retrieval without special consideration in the firewalls and NATs (e.g. an HTTP client on the UA can typically pull content from a server outside the NAT/firewall.).

A conscious separation of device, user, application and local network

profiles is made in this document. This is useful to provide features such as hotelling as well as securing or restricting user agent functionality. By maintaining this separation, a user may walk up to someone else's user agent and direct that user agent to get their profile data. In doing so the user agent can replace the previous user's profile data while still keeping the devices profile data that may be necessary for core functionality and communication described in this document. The local network profiles are relevant to a visiting device which gets plugged in to a foreign network. The concept of the local network providing profile data is useful to provide hotelling (described above) as well as local policy data that may constrain the user or device behavior relative to the local network. For example media types and codecs may be constrained to reflect the networks capabilities.

The separation of these profiles also enables the separation of the management of the profiles. The user profile may be managed by a profile delivery server operated by the user's ISP. The device profile may be delivered from a profile delivery server operated by the user's employer. The application profile may be delivered from the user's ASP. The local network profile may delivered by a WIFI hotspot service provider. Some interesting services and mobility applications are enabled with this separation of profiles.

A very high level data model is implied here with the separation of these four profile types. Each profile type requires a separate subscription to retrieve the profile. A loose hierarchy exists mostly for the purpose of boot strapping and discovery or formation of the profile URIs. No other meaning is implied by this hierarchy. However the profile format and data sets to be define outside this document, may define additional meaning to this hierarchy. In the boot strapping scenario, a device straight out of the box (software or hardware) does not know anything about it's user or local network. The one thing that is does know is it's instance id. So the hierarchy of the profiles exists as follows.

The instance id is used to form the URI for subscribing to the device profile. The device profile may contain a default user AOR for that device. The default user AOR may then be used to retrieve the user profile. Applications to be used on the device may be defined in the device and user profiles. The user's AOR is also used to retrieve any application profiles for that user. The local network profile is not referenced in any way from the device, user, application profiles. It is subscribed to and retrieved based upon a URI formed from the local network domain.

3. Profile Change Event Notification Package

This section defines a new SIP event package [RFC3265]. The purpose of this event package is to send to subscribers notification of content changes to the profile(s) of interest and to provide the location of the profile(s) via content indirection [I-D.ietf-sip-content-indirect-mech] or directly in the body of the NOTIFY. Frequently the profiles delivered to the user agent are much larger (e.g. several KB or even several MB) than the MTU of the network. These larger profiles will cause larger than normal SIP messages and consequently higher impact on the SIP servers and infrastructure. To avoid the higher impact and load on the SIP infrastructure, content indirection SHOULD be used if the profile is large enough to cause packet fragmentation over the transport protocol. The presence of the MIME type for content indirection [I-D.ietf-sip-content-indirect-mech] in the Accept header indicates that the user agent supports content indirection and that the profile delivery server SHOULD use content indirection. Similarly the content type for the differential notification of profile changes [I-D.ietf-simple-xcap-package] may be used in the Accept header to receive profile change deltas.

The MIME types or formats of profile to be delivered via this framework are to be defined in the documents that define the profile contents. These profile MIME types specified in the Accept header along with the profile types specified in the Event header parameter "profile-name" MAY be used to specify which profiles get delivered either directly or indirectly in the NOTIFY requests. As this event package does not specify the mandatory content type, this package is abstract. The profile definition documents will specify the mandatory content type to make a concrete event package.

3.1 Event Package Name

The name of this package is "sip-profile". This value appears in the Event header field present in SUBSCRIBE and NOTIFY requests for this package as defined in [RFC3265].

3.2 Event Package Parameters

This package defines the following new parameters for the event header: "profile-name", "vendor", "model", "version", "effective-by", "document", "app-id". The effective-by parameter is for use in NOTIFY requests only. The others are for use in the SUBSCRIBE request, but may be used in NOTIFY requests as well.

The "profile-name" parameter is used to indicate the token name of the profile type the user agent wishes to obtain data or URIs for and

to be notified of subsequent changes. Using a token in this parameter allows the URL semantics for retrieving the profiles to be opaque to the subscribing user agent. All it needs to know is the token value for this parameter. This document defines four logical types of profiles and their token names. The contents or format of the profiles is outside the scope of this document.

The four types of profiles define here are "device", "user", "application" and "local". Specifying "device" type profile(s) indicates the desire for the profile data (URI when content indirection is used) and change notification of the contents of the profile(s) that are specific to the device or user agent. Specifying "user" type profile indicates the desire for the profile data or URI to the profile(s) and change notification of the profile content for the user. Specifying "application" type profile indicates the desire for the profile data or URI to the profile(s) and change notification of the profile content for the user's applications. Specifying "local" type profile indicates the desire for profiles data or URI to the profile(s) specific to the local network. The device, user, application or local network is identified in the URI of the SUBSCRIBE request. The Accept header of the SUBSCRIBE request MUST include the MIME types for all profile content types that the subscribing user agent wishes to retrieve profiles or receive change notifications.

```
Profile-Name       =  "profile-name" HCOLON profile-value
profile-value      =  profile-types / token
profile-types      =  "device" / "user" / "application" / "local"
```

The "device", "user", "application" or "local" token in the profile-name parameter may represent a class or set of profile properties. As standards are defined for specific profile contents related to the user device or local network, it may be desirable to define additional tokens for the profile-name header. Also additional content types may be defined along with the profile formats that can be used in the Accept header of the SUBSCRIBE to filter or indicate what data sets of the profile are desired.

The rational for the separation of user, device and local network type profiles is provided in Section 2.3. It should be noted that any of the types may indicate that zero or more profiles or URIs are provided in the NOTIFY request. As discussed, a default user may be assigned to a device. The default user's AOR may in turn be used as the URI to SUBSCRIBE to the "user" and "application" profile types.

The data provided in the four types of profiles may overlap. As an example the codecs that a user prefers to use, the codecs that the

device supports (and the enterprise or device owner wishes to use), the codecs that the local network can support (and the network operator wishes to allow) all may overlap in how they are specified in the three corresponding profiles. This policy of merging the constraints across the multiple profile types can only unambiguously be defined along with the profile format and syntax. This is out of scope for this document.

The "vendor", "model" and "version" parameter values are tokens specified by the implementer of the user agent. These parameters are useful to the profile delivery server to affect the profiles provided. In some scenarios it is desirable to provide different profiles based upon these parameters. For example feature property X in a profile may work differently on two versions of user agent. This gives the profile deliver server the ability to compensate for or take advantage of the differences.

The "network-user" parameter is used when subscribing for local network profiles. If the value of the profile-name parameter is not "local", the "network-user" parameter has no defined meaning. If the user has special privileges beyond that of an anonymous user in the local network, the "network-user" parameter identifies the user to the local network. The value of this parameter is the user's address of record. The SUBSCRIBE server may authenticate the subscriber to verify this AOR.

The "effective-by" parameter in the Event header of the NOTIFY specifies the maximum number of seconds before the user agent MUST make the new profile effective. A value of 0 (zero) indicates that the user agent MUST make the profiles effective immediately (despite possible service interruptions). This gives the profile delivery server the power to control when the profile is effective. This may be important to resolve an emergency problem or disable a user agent immediately.

The "document" parameter is used to specify a relative URI for a specific profile document that the user agent wishes to retrieve and to receive change notification. This is particularly useful for profile content like XCAP [I-D.ietf-simple-xcap] where there is a well defined URL schema and the user agent knows the specific content that it wants. The "document" parameter value syntax is a quoted string. For more details on the use of this package with XCAP see Section 4.6.

The "app-id" parameter is only used when the "profile-name" parameter value is "application". The "app-id" indicates that the user agent wishes to retrieve the profile data or URI and change notification for the application profile data for the specific application

indicated in the value of the "app-id" parameter. The "app-id" parameter value is a token.

SUBSCRIBE request Event header examples:
Event: sip-profile;profile-name=device;
vendor=acme;model=Z100;version=1.2.3

Event: sip-profile;profile-name=
"http://example.com/services/user-profiles/users/freds.xml";
vendor=premier;model=trs8000;version=5.5

NOTIFY request Event header examples:
Event:sip-profile;effective-by=0

Event:sip-profile;effective-by=3600

3.3 SUBSCRIBE Bodies

This package defines no new use of the SUBSCRIBE request body. Future follow on documents may specify a filter-like mechanism using etags to minimize the delivery or notification of profiles where the user agent already has a current version.

3.4 Subscription Duration

As the presence (or lack of) a device or user agent it not very time critical to the functionality of the profile delivery server, it is recommended that default subscription duration be 86400 seconds (one day).

3.5 NOTIFY Bodies

The size of profile content is likely to be hundreds to several thousand bytes in size. Frequently even with very modest sized SDP bodies, SIP messages get fragmented causing problems for many user agents. For this reason if the Accept header of the SUBSCRIBE included the MIME type: message/external-body indicating support for content indirection the profile delivery server SHOULD use content indirection [I-D.ietf-sip-content-indirect-mech] in the NOTIFY body for providing the profiles.

When delivering profiles via content indirection the profile delivery server MUST include the Content-ID defined in [I-D.ietf-sip-content-indirect-mech] for each profile URL. This is to avoid unnecessary download of the profiles. Some user agents are not able to make a profile effective without rebooting or restarting.

Rebooting is something to be avoided on a user agent performing services such as telephony. In this way the Content-ID allows the user agent to avoid unnecessary interruption of service as well. The Content-Type MUST be specified for each URI.

> Initially user agent implementers may use a proprietary content type for the profiles retrieved from the URIs(s). This is a good first step towards easing the management of user agents. Standard profile contents, content type and formats will need to be defined for true interoperability of profile delivery. The specification of the content is out of the scope of this document.

Likewise the URL scheme used in the content indirection is outside the scope of this document. This document is agnostic to the URL schemes as the profile content may dictate what is required. It is expected that TFTP [RFC3617], FTP [??], HTTP [RFC2616], HTTPS [RFC2818], LDAP [RFC3377], XCAP [I-D.ietf-simple-xcap] and other URL schemes are supported by this package and framework.

3.6 Notifier processing of SUBSCRIBE requests

The general rules for processing SUBSCRIBE requests [RFC3265] apply to this package. If content indirection is used for delivering the profiles, the notifier does not need to authenticate the subscription as the profile content is not transported in the SUBSCRIBE or NOTIFY transaction messages. With content indirection only URLs are transported in the NOTIFY request which may be secured using the techniques in Section 6. If content indirection is not used, SIPS with SIP authentication SHOULD be used.

The behavior of the profile delivery server is left to the implementer. The profile delivery server may be as simple as a SIP SUBSCRIBE UAS and NOTIFY UAC front end to a simple HTTP server delivering static files that are hand edited. At the other extreme the profile delivery server can be part of a configuration management system that integrates with a corporate directory and IT system or carrier OSS, where the profiles are automatically generated. The design of this framework intentionally provides the flexibility of implementation from simple/cheap to complex/expensive.

If the user or device is not known to the profile delivery server, the implementer MAY accept the subscription or reject it. It is recommended that the implementer accept the subscription. It is useful for the profile delivery server to maintain the subscription as an administrator may add the user or device to the system, defining the profile contents. This allows the profile delivery server to immediately send a NOTIFY request with the profile URIs. If the profile delivery server does not accept the subscription from

an unknown user or device, the administer or user must manually provoke the user agent to reSUBSCRIBE. This may be difficult if the user agent and administrator are at different locations.

3.7 Notifier generation of NOTIFY requests

As in [RFC3265], the profile delivery server MUST always send a NOTIFY request upon accepting a subscription. If the device or user is unknown to the profile delivery server and it chooses to accept the subscription, the implementer has two choices. A NOTIFY MAY be sent with no body or content indirection containing the profile URI(s). Alternatively a NOTIFY MAY be sent with a body or content indirection containing URI(s) pointing to a default data set. The data sets provided may allow for only limited functionality of the user agent (e.g. a phone user agent with data to enable calls to help desk and emergency services.). This is an implementation and business policy decision for the profile delivery server.

If the URI in the SUBSCIRBE request is a known identity and provisioned with the requested profile type (i.e. as specified in the profile-name parameter), the profile delivery server SHOULD send a NOTIFY with profile data or content indirection (if the content type was included in the Accept header) containing the URI for the profile.

A user agent can provide hotelling by collecting a userËs AOR and credentials needed to SUBSCRIBE and retrieve the user's profiles. hotelling functionality is achieved by subscribing to the user's AOR and specifying the "user" profile type. This same mechanism can also be used to secure a user agent, requiring a user to login to enable functionality beyond the default userËs restricted functionality.

The profile delivery server MAY specify when the new profiles MUST be made effective by the user agent. By default the user agent makes the profiles effective as soon as it thinks that it is non-obtrusive. Profile changes SHOULD affect behavior on all new dialogs which are created after the notification, but may not be able to effect existing dialogs. However the profile delivery server MAY specify a maximum time in seconds (zero or more), in the effective-by event header parameter, by which the user agent MUST make the new profiles effective for all dialogs.

3.8 Subscriber processing of NOTIFY requests

The user agent subscribing to this event package MUST adhere to the NOTIFY request processing behavior specified in [RFC3265]. The user agent MUST make the profiles effective as specified in the NOTIFY request (see Section 3.7). The user agent SHOULD use one of the

techniques specified in Section 6 to securely retrieve the profiles.

3.9 Handling of forked requests

This event package allows the creation of only one dialog as a result of an initial SUBSCRIBE request. The techniques to achieve this are described in section 4.4.9 of [RFC3265].

3.10 Rate of notifications

It is anticipated that the rate of change for user and device profiles will be very infrequent (i.e. days or weeks apart). For this reason no throttling or minimum period between NOTIFY requests is specified for this package.

3.11 State Agents

State agents are not applicable to this event package.

3.12 Examples

Example SUBSCRIBE and NOTIFY request using content indirection:

```
SUBSCRIBE sip:ff00000036c5@example.com SIP/2.0
Event: sip-profile;profile-name=device;vendor=acme;
                        model=Z100;version=1.2.3
From: sip:ff00000036c5@acme.com;tag=1234
To: sip:ff00000036c5@acme.com;tag=abcd
Call-ID: 3573853342923422@10.1.1.44
CSeq: 2131 SUBSCRIBE
Contact: sip:ff00000036c5@10.1.1.44
Accept: message/external-body, application/z100-device-profile
Content-Length: 0


NOTIFY sip:ff00000036c5@10.1.1.44 SIP/2.0
Event: sip-profile;effective-by=3600
From: sip:ff00000036c5@acme.com;tag=abcd
To: sip:ff00000036c5@acme.com;tag=1234
Call-ID: 3573853342923422@10.1.1.44
CSeq: 321 NOTIFY
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=boundary42
Content-Length: ...

--boundary42
Content-Type: message/external-body;
    access-type="URL";
    expiration="Mon, 24 June 2002 09:00:00 GMT";
        URL="http://www.example.com/devices/ff00000036c5";
        size=1234

Content-Type: application/z100-device-profile
Content-ID: <39EHF78SA@example.com>

--boundary42--
```

3.13 Use of URIs to Retrieve State

The URI for the SUBSCRIBE request is formed differently depending upon which profile type the subscription is for. This allows the different profile types to be potentially managed by different profile delivery servers (perhaps even operated by different entities).

3.13.1 Device URIs

The URI for the "device" type profile is base upon the identity of the device. The device URI MUST be unique over time and space for

all devices and implementations. The instance id used as the user part of the device URI SHOULD remain the same for the lifetime of the user agent. The device URI is used to identify which profile is associated with a specific instance of a user agent.

> If the user agent were to change its device URI, the profile delivery server would loose its association between the profile and the device. This would also make it difficult for the profile delivery server to track user agents under profile management.

The URI for the device type profile should use a unique identifier as the user portion of the URI. The host and port portion of the URI as set to that of the domain or address of the profile deliver server which manages that user agent. A means of discovering the host and port portion is discussed in Section 4.1. Two approaches are suggested for constructing a unique identifier to be used in the user portion of the device URI.

> The MAC address of the device may be used if there will always be no more than one user agent using that MAC address over time (e.g. a dedicate telephone appliance). The MAC address may not be used if more than one user agent instance exists or use the same MAC address (e.g. multiple instances of a softphone may run on a general purpose computing device). The advantage of the MAC address is that many vendors put bar codes on the device with the actual MAC address on it. A bar code scanner is a convenient means of collecting the instance id for input and provisioning on the profile delivery server. If the MAC address is used, it is recommended that the MAC address is rendered in all lower case with no punctuation for consistency across implementations. For example a device managed by sipuaconfig.example.com using its MAC address to form the device URI might look like: sip:00df1e004cd0@sipuaconfig.example.com.
> For devices where there is no MAC address or the MAC address is not unique to an instance of a user agent (e.g. multiple softphones on a computer or a gateway with multiple logical user agents) it is recommended that a URN [RFC2141] is used as the user portion of the device URI. The approach to defining a user agent instance ID in for GRUU [I-D.ietf-sip-gruu] should be considered. When constructing the instance id the implementer should also consider that a human may need to manual enter the instance id to provision the device in the profile delivery server (i.e. longer strings are more error prone in data entry). When the URN is used as the user part of URI, it MUST be URL escaped. The ":" is not a legal character (without being escaped) in the user part of a name-addr. For example the instance ID: urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 would be escaped to look as follows in a URI:

sip:urn%3auuid%3af81d4fae-7dec-11d0-a765-00a0c91e6bf6@example.com.

3.13.2 User and Application URIs

The URI for the "user" and "application" type profiles is based upon the identity of the user. The user's address of record (AOR) is used as the URI in the SUBSCRIBE request. A new user agent or device may not know the user's AOR. The user's AOR may be obtained as part of a default user property in the device profile. Alternatively the user agent may prompt the user for an AOR to be used. This can provide a login and/or hotelling feature on the user agent.

3.13.3 Local Network URIs

The URI for the "local" type profile is based upon the identity of the local network. When subscribing to the local network profile, the use part of the URI is "anonymous". The host and port part of the URI is the local network name/domain. The discovery of the local network name or domain is discussed in Section 4.1. The user agent may provide the user's AOR as the value to the "network-user" event header parameter. This is useful if the user has privileges in the local network beyond those of the default user. The profile delivery server SHOULD authenticate the user before providing the profile if additional privileges are granted. Example URI: sip:ananymous@example.com

4. Profile Delivery Framework Details

The following describes how different functional steps of the profile delivery framework work. Also described here is how the event package defined in this document provides the enrollment and notification functions within the framework.

4.1 Discovery of Subscription URI

The discover approach varies depending upon which profile type URI is to be discovered. The order of discover is important in the boot strapping situation as user agent may not have any information provisioned. The local network profile should be discovered first as it may contain key information such as how to traverse a NAT/firewall to get to outside services (e.g. the user's profile delivery server). The device profile URI should be discovered next. The device profile may contain the default user's AOR. The user and application profile subscription URI's are discovered last.

4.1.1 Discovery of Local Network URI

The "discovered" host for the "local" profile subscription URI is the

local IP network domain for the user agent, either provisioned as part of the device's static network configuration or discovered via DHCP. The local network profile subscription URI should not be cached as the user agent may be move from one local network to the other. The user agent should perform the local network discovery every time it starts up or network connectivity is regained.

4.1.2 Discovery of Device URI

The discovery function is needed to bootstrap user agents to the point of knowing where to enroll with the profile delivery server. Section 3.13.1 describes how to form the device URI used to send the SUBSCRIBE request for enrollment. However the bootstrapping problem for the user agent (out of the box) is what to use for the host and port in the device URI. Due to the wide variation of environments in which the enrolling user agent may reside (e.g. behind residential router, enterprise LAN, WIFI hotspot, ISP, dialup modem) and the limited control that the administrator of the profile delivery server (e.g. enterprise, service provider) may have over that environment, no single discovery mechanism works everywhere. Therefore a number of mechanisms SHOULD be tried in the specified order: SIP DHCP option [RFC3361], SIP DNS SRV [RFC3263], DNS A record and manual. The user agent may be preprovisioned with the host and port (e.g. service providers may preprovision a device before sending it to a subscriber) in which case this discovery mechanism is not needed. Before performing the discover steps, the user agent SHOULD provide a means to skip the discovery stage and manually enter the device URI host and port. In addition the user agent SHOULD allow the user to accept or reject the discovered host and port, in case an alternate to the discovered host and port are desired.

1. The first discovery mechanism that SHOULD be tried is to construct the device SUBSCRIBE URI, as described in Section 3.13.1, is to use the host and port of the out bound proxy discovered by the SIP DHCP option as described in [RFC3361]. If the SIP DHCP option is not provided in the DHCP response; or no SIP response is received for the SUBSCRIBE request; or a SIP failure response other than for authorization is received for the SUBSCRIBE request to the sip-profile event, the next discovery mechanism SHOULD be tried.
2. The local IP network domain for the user agent, either configured or discovered via DHCP, should be used with the technique in [RFC3263] to obtain a host and port to use in the SUBSCRIBE URI. If no SIP response or a SIP failure response other than for authorization is received for the SUBSCRIBE request to the sip-profile event, the next discovery mechanism SHOULD be tried.
3. The fully qualified host name constructed using the host name "sipuaconfig" and concatenated with the local IP network domain

(as provided via DHCP or provisioned) should be tried next using the technique in [RFC3263] to obtain a host and port to use in the SUBSCRIBE URI. If no SIP response or a SIP failure response other than for authorization is received for the SUBSCRIBE request to the sip-profile event, the next discovery mechanism SHOULD be tried.

4. If all other discovery techniques fail, the user agent MUST provide a manual means for the user to enter the host and port used to construct the SUBSCRIBE URI.

Once a user agent has successfully discovered, enrolled, received a NOTIFY response with profile data or URI(s), the user agent SHOULD cache the device profile SUBCRIBE URI to avoid having to rediscover the profile delivery server again in the future. The user agent SHOULD NOT cache the SUBSCRIBE URI until it receives a NOTIFY with profile data or URI(s). The reason for this is that a profile delivery server may send 202 responses to SUBSCRIBE requests and NOTIFY responses to unknown user agent (see Section 3.6) with no URIs. Until the profile delivery server has sent a NOTIFY request with profile data or URI(s), it has not agreed to provide profiles.

To illustrate why the user agent should not cache the device profile SUBSCRIBE URI until profile data or URI(s) are provided in the NOTIFY, consider the following example: a user agent running on a laptop plugged into a visited LAN in which a foreign profile delivery server is discovered. The profile delivery server never provides profile URIs in the NOTIFY request as it is not provisioned to accept the user agent. The user then takes the laptop to their enterprise LAN. If the user agent cached the SUBSCRIBE URI from the visited LAN (which did not provide profiles), when subsequently placed in the enterprise LAN which is provisioned to provide profiles to the user agent, the user agent would not attempt to discover the profile delivery server.

4.1.3 Discovery of User and Application URI

The default user's AOR from the device profile (if provided) may then be used to subscribe to the "user" and "application" profiles. Alternatively the user's AOR to be used for the "user" and application" subscription URI, may be "discovered" manually by prompting the user. This "discovered" URI for the user and application profile subscription may be cached.

4.2 Enrollment with Profile Server

Enrollment is accomplished by subscribing to the event package described in Section 3. The enrollment process is useful to the profile delivery server as it makes the server aware of user agents

to which it may delivery profiles (those user agents the profile delivery server is provisioned to provide profiles to; those present that the server may be provide profiles in the future; and those that the server can automatically provide default profiles). It is an implementation choice and business policy as to whether the profile delivery server provides profiles to user agents that it is not explicitly provisioned to do so. However the profile server SHOULD accept (with 2xx response) SUBSCRIBE requests from any user agent as explained in Section 3.5.

4.3 Notification of Profile Changes

The NOTIFY request in the sip-profile event package serves two purposes. First it provides the user agent with a means to obtain the profile directly data or via URI(s) for desired profiles without requiring the end user to manually enter them. It also provides the means for the profile delivery server to notify the user agent that the content of the profiles have changed and should be made effective. Optionally the differential changes may be obtained by including the content-type defined in [I-D.ietf-simple-xcap-package] in the Accept header of the SUBSCRIBE request.

4.4 Retrieval of Profile Data

The user agent retrieves its needed profile(s) directly or via the URI(s) provided in the NOTIFY request as specified in Section 3.5. The profile delivery server SHOULD secure the content of the profiles using one of the techniques described in Section 6. The user agent SHOULD make the new profiles effective in the timeframe described in Section 3.2.

The contents of the profiles SHOULD be cached by the user agent. This it to avoid the situation where the content delivery server is not available, leaving the user agent non-functional.

4.5 Upload of Profile Changes

The user agent or other service MAY push changes up to the profile delivery server using the technique appropriate to the profile's URL scheme (e.g. HTTP PUT method, FTP put command). The technique for pushing incremental or atomic changes MUST be described by the specific profile data framework. A means for pushing changes up into the profile delivery server for XCAP is defined in [I-D.ietf-simple-xcap].

4.6 Usage of XCAP with the Profile Package

This framework allows for the usage of several different protocols

for the retrieval of profiles. One protocol which is suitable is XCAP [I-D.ietf-simple-xcap], which allows for HTTP URIs to represent XML documents, elements and attributes. XCAP defines a specific hierarchy for how documents are organized. As a result, it is necessary to discuss how that organization relates to the rough data model presented here.

When a user or device enrolls with a SUBSCRIBE request, the request will contain some kind of identifying information for that user or device. This identity is mapped to an XCAP User ID (XUID) based on an implementation specific mapping. The "profile-name" along with the "app-id" Event header parameters specify the specific XCAP application usage.

In particular, when the "profile-name" is "application", the "app-id" contains the XCAP Application Unique ID (AUID). When the "profile-name" is application, but the "app-id" parameter is absent, this specifies that the user wishes to SUBSCRIBE to all documents for all application usages associated with the user in the request-uri. This provides a convenient way for a single subscription to be used to obtain all application data. The XCAP root is determined by a local mapping.

When the "profile-name" is "device", or "user" or "local-network", this maps to an AUID and document selector for representing device, user and local-network data, respectively. The mapping is a matter of local policy. This allows different providers to use different XCAP application usages and document schemas for representing these profiles, without having to configure the device with the specific AUID which is being used.

Furthermore, when the "document" attribute is present, it identifies a specific document that is being requested. If the "profile-name" is "application", the "app-id" MUST be present as well. The "document" attribute then specifies a relative path reference. Its first path segment is either "global", specifying global data, or "user", specifying user data for the user in the request URI. The next path segment identifies the path in the global directory or the user's home directory.

For example, consider a phone with an instance ID of urn:uuid:00000000-0000-0000-0000-0003968cf920. To obtain its device profile, it would generate a SUBSCRIBE that looks like this:

```
SUBSCRIBE
sip:urn%3auuid%3a00000000-0000-0000-0000-0003968cf920@example.com
Event: sip-profile;profile-name=device
```


If the profile data is stored in an XCAP server, the server would the "device" profile to an application usage and document selector based on local policy. If this mapping specifies the AUID "vendor2-device-data" and a document called "index" within the user directory, the corresponding HTTP URI for the document is:

```
http://xcap.example.com/root/vendor2-device-data/users/
urn%3auuid%3a00000000-0000-0000-0000-0003968cf920/index
```

and indeed, if a content indirection is returned in a NOTIFY, the URL would equal this.

That user profile might specify the user identity (as a SIP AOR) and their application-usages. From that, the device can enroll to learn about its application data. To learn about all of the data:

```
SUBSCRIBE sip:user-aor@example.com SIP/2.0
Event: sip-profile;profile-name=application
```

The server would map the request URI to an XUI (user-aor, for example) and the xcap root based on local policy. If there are two AUIDs, "resource-lists" [I-D.ietf-simple-xcap-list-usage] and "rls-services" [I-D.ietf-simple-xcap-list-usage], this would result in a subscription to all documents within:

```
http://xcap.example.com/root/rls-services/users/user-aor
http://xcap.example.com/root/resource-lists/users/user-aor
```

The user would not be subscribed to the global data for these two application usages, since that data is not important for users.

However, the user/device could be made aware that it needs to subscribe to a specific document. In that case, its subscribe would look like:

```
SUBSCRIBE sip:user-aor@example.com SIP/2.0
Event: sip-profile;profile-name=application;app-id=resource-lists
  ;document="global/index"
```

this would result in a subscription to the single global document for resource-lists.

In some cases, these subscriptions are to a multiplicity of documents. In that case, the notification format will need to be one

which can indicate what document has changed. This includes content indirection, but also the xcap diff format [I-D.ietf-simple-xcap-package].

5. IANA Considerations

There are several IANA considerations associated with this specification.

5.1 SIP Event Package

This specification registers a new event package as defined in [RFC3265]. The following information required for this registration:

Package Name: sip-profile
Package or Template-Package: This is a package
Published Document: RFC XXXX (Note to RFC Editor: Please fill in XXXX with the RFC number of this specification).
Person to Contact: Daniel Petrie dpetrie@pingtel.com
New event header parameters: profile-name, vendor, model, version, effective-by, document, app-id

6. Security Considerations

Profiles may contain sensitive data such as user credentials. The protection of this data depends upon how the data is delivered. If the data is delivered in the NOTIFY body, SIP authentication MUST be used for SUBSCRIPTION and SIPS and/or S/MIME MAY be use to encrypt the data. If the data is provided via content indirection, SIP authentication is not necessary for the SUBSCRIBE request. With content indirection the data is protected via the authentication, authorization and encryption mechanisms provided by the profile URL scheme. Use of the URL scheme security mechanisms via content indirection simplifies the security solution as the SIP event package does not need to authenticate, authorize or protect the contents of the SIP messages. Effectively the profile delivery server can safely provide profile URI(s) to anyone. The profile content is protected via the URL scheme transport mechanisms for authentication, authorization and encryption (e.g. via HTTPS). HTTPS provides two possible mechanisms for authentication: 1) the device may have a certificate that the profile deliver server can request in the TLS setup; or 2) the profile deliver server may use HTTP authentication [RFC2617] with the device or users credentials.

6.1 Symmetric Encryption of Profile Data

If the transport for the URL scheme used for content indirection does not provide authentication, authorization or encryption, a technique to provide this is to encrypt the profiles on the content delivery

server using a symmetric encryption algorithm using a shared key. The encrypted profiles are delivered by the content delivery server via the URIs provided in the NOTIFY requests. Using this technique the profile delivery server does not need to provide authentication or authorization for the retrieval as the profiles are obscured. The user agent must obtain the username and password from the user or other out of band means to generate the key and decrypt the profiles.

7. Change History

Many thanks to those who contributed and commented on the many iterations of this document. Detailed input was provided by Jonathan Rosenberg from Dynamicsoft, Henning Schulzrinne from Columbia U., Cullen Jennings from Cisco, Rohan Mahy from Cisco, Rich Schaaf from Pingtel, Volker Hilt from Bell Labs.

7.1 Changes from draft-ietf-sipping-config-framework-03.txt

Incorporated changes to better support the requirements for the use of this event package with XCAP and SIMPLE so that we can have one package (i.e. simple-xcap-package now defines a content type not a package). Added an additional profile type: application. Added document and app-id Event header parameters in support of the application profile. Define a loose high level data model or relationship between the four profile types. Tried to edit and fix the confusing and ambiguous sections related to URI formation and discovery for the different profile types. Better describe the importance of uniqueness for the instance id which is used in the user part of the device URI.

7.2 Changes from draft-ietf-sipping-config-framework-02.txt

Added the concept of the local network as a source of profile data. There are now three separate logical sources for profile data: user, device and local network. Each of these requires a separate subscription to obtain.

7.3 Changes from draft-ietf-sipping-config-framework-01.txt

Changed the name of the profile-type event parameter to profile-name. Also allow the profile-name parameter to be either a token or an explicit URI.

Allow content indirection to be optional. Clarified the use of the Accept header to indicate how the profile is to be delivered.

Added some content to the Iana section.

7.4 Changes from draft-ietf-sipping-config-framework-00.txt

This version of the document was entirely restructured and re-written from the previous version as it had been micro edited too much.

All of the aspects of defining the event package are now organized in one section and is believed to be complete and up to date with [RFC3265].

The URI used to subscribe to the event package is now either the user or device address or record.

The user agent information (vendor, model, MAC and serial number) are now provided as event header parameters.

Added a mechanism to force profile changes to be make effective by the user agent in a specified maximum period of time.

Changed the name of the event package from sip-config to sip-profile

Three high level security approaches are now specified.

7.5 Changes from draft-petrie-sipping-config-framework-00.txt

Changed name to reflect SIPPING work group item

Synchronized with changes to SIP DHCP [RFC3361], SIP [RFC3261] and [RFC3263], SIP Events [RFC3265] and content indirection [I-D.ietf-sip-content-indirect-mech]

Moved the device identity parameters from the From field parameters to User-Agent header parameters.

Many thanks to Rich Schaaf of Pingtel, Cullen Jennings of Cisco and Adam Roach of Dyamicsoft for the great comments and input.

7.6 Changes from draft-petrie-sip-config-framework-01.txt

Changed the name as this belongs in the SIPPING work group.

Minor edits

7.7 Changes from draft-petrie-sip-config-framework-00.txt

Split the enrollment into a single SUBSCRIBE dialog for each profile. The 00 draft sent a single SUBSCRIBE listing all of the desired. These have been split so that each enrollment can be routed differently. As there is a concept of device specific and user

specific profiles, these may also be managed on separate servers. For instance in a roaming situation the device might get its profile data from a local server which knows the LAN specific profile data. At the same time the user specific profiles might come from the user's home environment profile delivery server.

Removed the Config-Expires header as it is largely superfluous with the SUBSCRIBE Expires header.

Eliminated some of the complexity in the discovery mechanism.

Suggest caching information discovered about a profile delivery server to avoid an avalanche problem when a whole building full of devices powers up.

Added the User-Profile From header field parameter so that the device can request a user specific profile for a user that is different from the device's default user.

8 References

[I-D.ietf-simple-xcap]
Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-02 (work in progress), February 2004.

[I-D.ietf-simple-xcap-list-usage]
Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[I-D.ietf-simple-xcap-package]
Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Modification Events for the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Managed Documents", draft-ietf-simple-xcap-package-01 (work in progress), February 2004.

[I-D.ietf-sip-content-indirect-mech]
Olson, S., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", draft-ietf-sip-content-indirect-mech-03 (work in progress), June 2003.

[I-D.ietf-sip-gruu]
Rosenberg, J., "Obtaining and Using Globally Routable User

Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-02 (work in progress), July 2004.

[I-D.ietf-sipping-ua-prof-framewk-reqs] Petrie, D. and C. Jennings, "Requirements for SIP User Agent Profile Delivery Framework", draft-ietf-sipping-ua-prof-framewk-reqs-00 (work in progress), March 2003.

[I-D.petrie-sipping-profile-datasets] Petrie, D., "A Schema for Session Initiation Protocol User Agent Profile Data Sets", draft-petrie-sipping-profile-datasets-00 (work in progress), July 2004.

[I-D.sinnreich-sipdev-req] Butcher, I., Lass, S., Petrie, D., Sinnreich, H. and C. Stredicke, "SIP Telephony Device Requirements and Configuration", draft-sinnreich-sipdev-req-04 (work in progress), July 2004.

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3361] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers", RFC 3361, August 2002.

[RFC3377] Hodges, J. and R. Morgan, "Lightweight Directory Access Protocol (v3): Technical Specification", RFC 3377, September 2002.

[RFC3617] Lear, E., "Uniform Resource Identifier (URI) Scheme and Applicability Statement for the Trivial File Transfer Protocol (TFTP)", RFC 3617, October 2003.

[W3C.REC-xml-names11-20040204] Layman, A., Tobin, R., Bray, T. and D. Hollander, "Namespaces in XML 1.1", W3C REC REC-xml-names11-20040204, February 2004.

Author's Address

Daniel Petrie
Pingtel Corp.
400 W. Cummings Park
Suite 2200
Woburn, MA 01801
US

Phone: "Dan Petrie (+1 781 938 5306)"<sip:dpetrie@pingtel.com>
EMail: dpetrie@pingtel.com
URI: http://www.pingtel.com/

Appendix A. Acknowledgments

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgment

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

SIPPING
Internet-Draft
Expires: August 13, 2004

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia University
R. Mahy, Ed.
Cisco Systems, Inc.
February 13, 2004

An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)

draft-ietf-sipping-dialog-package-04.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 13, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document defines a dialog event package for the SIP Events architecture, along with a data format used in notifications for this package. The dialog package allows users to subscribe to another user, an receive notifications about the changes in state of INVITE initiated dialogs that the user is involved in.

Table of Contents

1. Introduction

The SIP Events framework [1] defines general mechanisms for subscription to, and notification of, events within SIP networks. It introduces the notion of a package, which is a specific "instantiation" of the events mechanism for a well-defined set of events. Packages have been defined for user presence [14], watcher information [15], and message waiting indicators [16], amongst others. Here, we define an event package for INVITE initiated dialogs. Dialogs refer to the SIP relationship established between two SIP peers [2]. Dialogs can be created by many methods, although RFC 3261 defines only one - the INVITE method. RFC 3265 defines the SUBSCRIBE and NOTIFY methods, which also create dialogs. However, the usage of this package to model transitions in the state of those dialogs is out of the scope of this specification.

There are a variety of applications enabled through the knowledge of INVITE dialog state. Some examples include:

Automatic Callback: In this basic Public Switched Telephone Network (PSTN) application, user A calls user B. User B is busy. User A would like to get a callback when user B hangs up. When B hangs up, user A's phone rings. When A picks it up, they here ringing, and are being connected to B. To implement this with SIP, a mechanism is required for B to receive a notification when the dialogs at A are complete.

Presence-Enabled Conferencing: In this application, a user A wishes to set up a conference call with users B and C. Rather than scheduling it, it is to be created automatically when A, B and C are all available. To do this, the server providing the application would like to know whether A, B and C are "online", not idle, and not in a phone call. Determining whether or not A, B and C are in calls can be done in two ways. In the first, the server acts as a call stateful proxy for users A, B and C, and therefore knows their call state. This won't always be possible, however, and it introduces scalability, reliability, and operational complexities. Rather, the server would subscriber to the dialog state of those users, and receive notifications as it changes. This enables the application to be provided in a distributed way; the server need not reside in the same domain as the users.

IM Conference Alerts: In this application, a user can get an IM sent to their phone whenever someone joins a conference that the phone is involved in. The IM alerts are generated by an application separate from the conference server.

In general, the dialog package allows for construction of distributed applications, where the application requires information on dialog state, but is not co-resident with the end user on which that state resides.

2. Terminology

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 [9] and indicate requirement levels for compliant implementations.

3. Dialog Event Package

This section provides the details for defining a SIP Events package, as specified by [1].

3.1 Event Package Name

The name of this event package is "dialog". This package name is carried in the Event and Allow-Events header, as defined in [1].

3.2 Event Package Parameters

This package defines four Event Package parameters. They are call-id, to-tag, from-tag, and include-session-description. If a subscription to a specific dialog is requested, all of the first three of these parameters MUST be present. They identify the dialog that is being subscribed to. The to-tag is matched against the local tag, the from-tag is matched against the remote tag, and the call-id is matched against the Call-ID. The include-session-description parameter indicates if the subscriber would like to receive the session descriptions associated with the subscribed dialog or dialogs.

It is also possible to subscribe to the set of dialogs created as a result of a single INVITE sent by a UAC. In that case, the call-id and to-tag MUST be present. The to-tag is matched against the local tag, and the call-id is matched against the Call-ID.

The ABNF for these parameters is shown below. It refers to many constructions from the ABNF of RFC3261, such as EQUAL, DQUOTE, and token.

```
call-id     =  "call-id" EQUAL ( token / DQUOTE callid DQUOTE )
                 ;; NOTE: any DQUOTEs inside callid MUST be escaped!
from-tag    =  "from-tag" EQUAL token
to-tag      =  "to-tag" EQUAL token
```

with-sessd = "include-session-description"

Any callids which contain embedded double-quotes MUST escape those double-quotes using the backslash-quoting mechanism. Note that the call-id parameter may need to be expressed as a quoted string. This is because the ABNF for callid and word (which is used by callid) allow for some characters (such as "@", "[", and ":") which are not allowed within a token.

3.3 SUBSCRIBE Bodies

A SUBSCRIBE for a dialog package MAY contain a body. This body defines a filter to apply to the subscription. Filter documents are not specified in this document, and at the time of writing, are expected to be the subject of future standardization activity.

A SUBSCRIBE for a dialog package MAY be sent without a body. This implies the default subscription filtering policy. The default policy is:

- If the Event header field contained dialog identifiers, notifications are generated every time there is a change in the state of any matching dialogs for the user identified in the request URI of the SUBSCRIBE.

- If there were no dialog identifiers in the Event header field, notifications are generated every time there is any change in the state of any dialogs for the user identified in the request URI of the SUBSCRIBE with the following exceptions. If the target (Contact) URI of a subscriber is equivalent to the remote target URI of a specific dialog, then the dialog element for that dialog is suppressed for that subscriber. (The subscriber is already a party in the dialog directly, so these notifications are superfluous.) If no dialogs remain after supressing dialogs, the entire notification to that subscriber is supressed and the version number in the dialog-info element is not incremented for that subscriber. Implicit filtering for one subscriber does not affect notifications to other subscribers.

- Notifications do not normally contain full state; rather, they only indicate the state of the dialog whose state has changed. The exceptions are a NOTIFY sent in response to a SUBSCRIBE, and a NOTIFY that contains no dialog elements. These NOTIFYs contain the complete view of dialog state.

- The notifications contain the identities of the participants in the dialog, the target URIs, and the dialog identifiers. Session descriptions are not included normally unless explicitly requested

and/or explicitly authorized.

3.4 Subscription Duration

Dialog state changes fairly quickly; once established, a typical phone call lasts a few minutes (this is different for other session types, of course). However, the interval between new calls is typically infrequent. As such, we arbitrarily choose a default duration of one hour. Clients SHOULD specify an explicit duration.

There are two distinct use cases for dialog state. The first is when a subscriber is interested in the state of a specific dialog or dialogs (and they are authorized to find out about just the state of those dialogs). In that case, when the dialogs terminate, so too does the subscription. In these cases, the value of the subscription duration is largely irrelevant, and SHOULD be longer than the typical duration of a dialog, about two hours would cover most dialogs.

In another case, a subscriber is interested in the state of all dialogs for a specific user. In these cases, a shorter interval makes more sense. The default is one hour for these subscriptions.

3.5 NOTIFY Bodies

As described in RFC 3265 [1], the NOTIFY message will contain bodies that describe the state of the subscribed resource. This body is in a format listed in the Accept header field of the SUBSCRIBE, or a package-specific default if the Accept header field was omitted from the SUBSCRIBE.

In this event package, the body of the notification contains a dialog information document. This document describes the state of one or more dialogs associated with the subscribed resource. All subscribers and notifiers MUST support the "application/dialog-info+xml" data format described in Section 4. The subscribe request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/dialog-info+xml". If the header field is present, it MUST include "application/dialog-info+xml", and MAY include any other types capable of representing dialog state.

Of course, the notifications generated by the server MUST be in one of the formats specified in the Accept header field in the SUBSCRIBE request.

3.6 Notifier Processing of SUBSCRIBE Requests

The dialog information for a user contains sensitive information.

Therefore, all subscriptions SHOULD be authenticated and then authorized before approval. All implementors of this package MUST support the digest authentication mechanism as a baseline. Authorization policy is at the discretion of the administrator, as always. However, a few recommendations can be made.

It is RECOMMENDED that, if the policy of user B is that user A is allowed to call them, dialog subscriptions from user A be allowed. However, the information provided in the notifications does not contain any dialog identification information; merely an indication of whether the user is in at least one call, or not. Specifically, they should not be able to find out any more information than if they sent an INVITE. (This concept of a "virtual" dialog is discussed more in Section 3.7.2, and an example of such a notification body is shown below.)

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="0" state="full"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8">
    <state>confirmed</state>
  </dialog>
</dialog-info>
```

It is RECOMMENDED that if a user agent registers with the address-of-record X, that this user agent authorize subscriptions that come from any entity that can authenticate itself as X. Complete information on the dialog state SHOULD be sent in this case. This authorization behavior allows a group of devices representing a single user to all become aware of each other's state. This is useful for applications such as single-line-extension.

> Note that many implementations of "shared-lines" have a feature which allows details of calls on a shared address-of-record to be made private. This is a completely reasonable authorization policy which could result in notifications which contain only the id attribute of the dialog element and the state element when shared-line privacy is requested, and notifications with more complete information when shared-line privacy is not requested.

3.7 Notifier Generation of NOTIFY Requests

Notifications are generated for the dialog package when an INVITE request is sent, when a new dialog comes into existence at a UA, or when the state or characteristics of an existing dialog changes. Therefore, a model of dialog state is needed in order to determine precisely when to send notifications, and what their content should

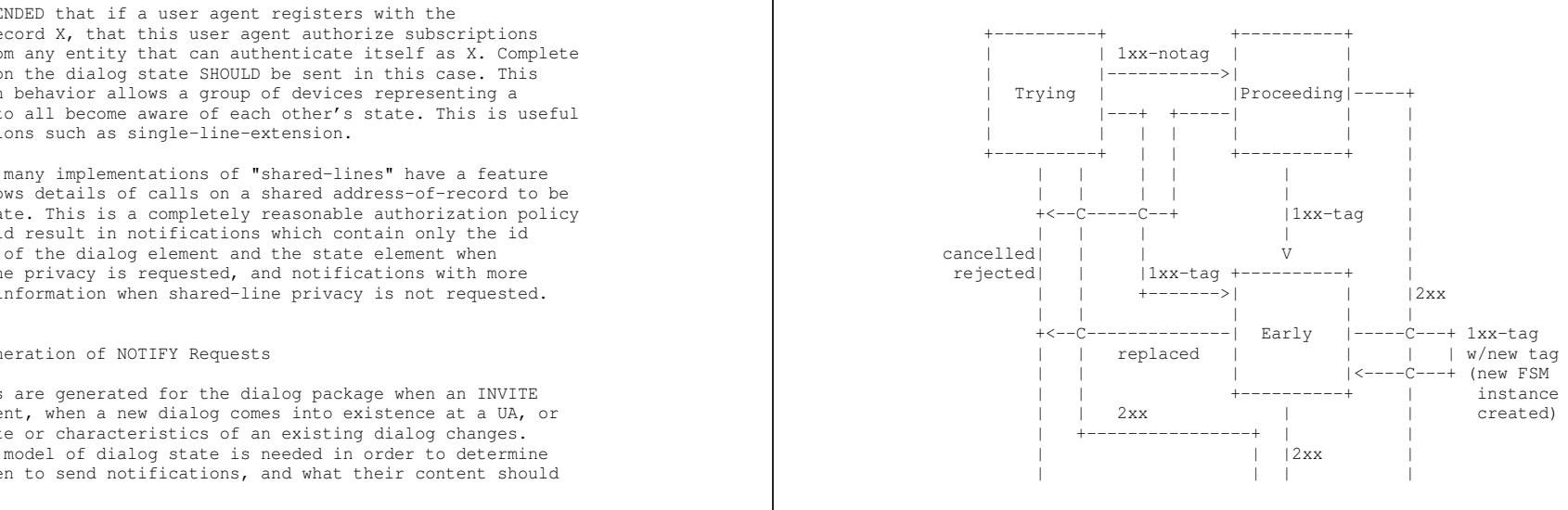
be. The SIP specification has a reasonably well defined lifecycle for dialogs. However, it is not explicitly modelled. This specification provides an explicit model of dialog state through a finite state machine.

It is RECOMMENDED that NOTIFY requests only contain information on the dialogs whose state or participation information has changed. However, if a notifier receives a SUBSCRIBE request, the triggered NOTIFY SHOULD contain the state of all dialogs that the subscriber is authorized to see.

3.7.1 The Dialog State Machine

Modelling of dialog state is complicated by two factors. The first is forking, which can cause a single INVITE to generate many dialogs at a UAC. The second is the differing views of state at the UAC and UAS. We have chosen to handle the first issue by extending the dialog FSM to include the states between transmission of the INVITE and the creation of actual dialogs through receipt of 1xx and 2xx responses. As a result, this specification supports the notion of dialog state for dialogs before they are fully instantiated.

We have also chosen to use a single FSM for both UAC and UAS.



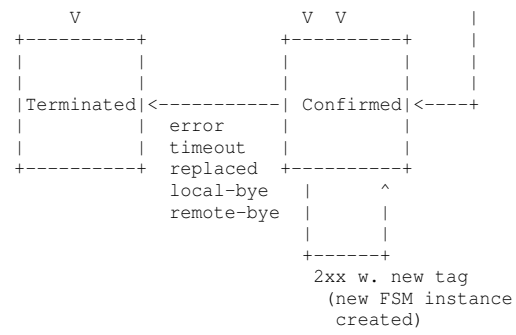


Figure 3

The FSM for dialog state is shown in Figure 3. The FSM is best understood by considering the UAC and UAS cases separately.

The FSM is created in the "trying" state when the UAC sends an INVITE request. Upon receipt of a 1xx without a tag, the FSM transitions to the "proceeding" state. Note that there is no actual dialog yet, as defined by the SIP specification. However, there is a "half-dialog", in the sense that two of the three components of the dialog ID are known (the call identifier and local tag). If a 1xx with a tag is received, the FSM transitions to the early state. The full dialog identifier is now defined. Had a 2xx been received, the FSM would have transitioned to the "confirmed" state.

If, after transitioning to the "early" or "confirmed" states, the UAC receives another 1xx or 2xx respectively with a different tag, another instance of the FSM is created, initialized into the "early" or "confirmed" state respectively. The benefit of this approach is that there will be a single FSM representing the entire state of the invitation and resulting dialog when dealing with the common case of no forking.

If the UAC should send a CANCEL, and then subsequently receive a 487 to its INVITE transaction, all FSMs spawned from that INVITE transition to the "terminated" state with the event "cancelled". If the UAC receives a new invitation (with a Replaces [13] header) which replaces the current Early or Confirmed dialog, all INVITE transactions spawned from the replaced invitation transition to the "terminated" state with the event "replaced". If the INVITE transaction terminates with a non-2xx response for any other reason, all FSMs spawned from that INVITE transition to the terminated state

with the event "rejected".

Once in the confirmed state, the call is active. It can transition to the terminated state if the UAC sends a BYE or receives a BYE (corresponding to the "local-bye" and "remote-bye" events as appropriate), if a mid-dialog request generates a 481 or 408 response (corresponding to the "error" event), or a mid-dialog request generates no response (corresponding to the "timeout" event).

From the perspective of the UAS, when an INVITE is received, the FSM is created in the "trying" state. If it sends a 1xx without a tag, the FSM transitions to the "proceeding" state. If a 1xx is sent with a tag, the FSM transitions to the "early" state, and if a 2xx is sent, it transitions to the "confirmed" state. If the UAS should receive a CANCEL request and then generate a 487 response to the INVITE (which can occur in the proceeding and early states), the FSM transitions to the terminated state with the event "cancelled". If the UAS should generate any other non-2xx final response to the INVITE request, the FSM transitions to the terminated state with the event "rejected". If the UAS receives a new invitation (with a Replaces [13] header) which replaces the current Confirmed dialog, the replaced invitation transition transitions to the "terminated" state with the event "replaced". Once in the "confirmed" state, the other transitions to the "terminated" state occur for the same reasons they do in the case of UAC.

> There should never be a transition from the "trying" state to the "terminated" state with the event "cancelled", since the SIP specification prohibits transmission of CANCEL until a provisional response is received. However, this transition is defined in the FSM just to unify the transitions from trying, proceeding, and early to the terminated state.

3.7.2 Applying the state machine

The notifier MAY generate a NOTIFY request on any event transition of the FSM. Whether it does or not is policy dependent. However, some general guidelines are provided.

When the subscriber is unauthenticated, or is authenticated, but represents a third party with no specific authorization policies, it is RECOMMENDED that subscriptions to an individual dialog, or to a specific set of dialogs, is forbidden. Only subscriptions to all dialogs (i.e., there are no dialog identifiers in the Event header field) are permitted. In that case, actual dialog states across all dialogs will not be reported. Rather, a single "virtual" dialog FSM be used, and event transitions on that FSM be reported.

If there is any dialog at the UA whose state is "confirmed", the virtual FSM is in the "confirmed" state. If there are no dialogs at the UA in the confirmed state, but there is at least one in the "early" state, the virtual FSM is in the "early" or "confirmed" state. If there are no dialogs in the confirmed or early states, but there is at least one in the "proceeding" state, the virtual FSM is in the "proceeding", "early" or "confirmed" state. If there are no dialogs in the confirmed, early, or proceeding states, but there is at least one in the "trying" state, the virtual FSM is in the "trying", "proceeding", "early" or "confirmed" state. The choice about which state to use depends on whether the UA wishes to let unknown users know that their phone is ringing, as opposed to in an active call.

It is RECOMMENDED that, in the absence of any preference, "confirmed" is used in all cases (as shown in the example in Section 3.6. Furthermore, it is RECOMMENDED that the notifications of changes in the virtual FSM machine not convey any information except the state of the FSM and its event transitions - no dialog identifiers (which are ill-defined in this model in any case). The use of this virtual FSM allows for minimal information to be conveyed. A subscriber cannot know how many calls are in progress, or with whom, just that there exists a call. This is the same information they would receive if they simply sent an INVITE to the user instead; a 486 response would indicate that they are on a call.

When the subscriber is authenticated, and has authenticated itself with the same address-of-record that the UA itself uses, if no explicit authorization policy is defined, it is RECOMMENDED that all state transitions on dialogs that have been subscribed to (which is either all of them, if no dialog identifiers were present in the Event header field, or a specific set of them identified by the Event header field parameters) be reported, along with complete dialog IDs.

The notifier MAY generate a NOTIFY request on any change in the characteristics associated with the dialog. Since these include Contact URIs, Contact parameters and session descriptions, receipt of re-INVITEs and UPDATE requests [3] which modify this information MAY trigger notifications.

3.8 Subscriber Processing of NOTIFY Requests

The SIP Events framework expects packages to specify how a subscriber processes NOTIFY requests in any package specific ways, and in particular, how it uses the NOTIFY requests to contruct a coherent view of the state of the subscribed resource.

Typically, the NOTIFY for the dialog package will only contain

information about those dialogs whose state has changed. To construct a coherent view of the total state of all dialogs, a subscriber to the dialog package will need to combine NOTIFYs received over time.

Notifications within this package can convey partial information; that is, they can indicate information about a subset of the state associated with the subscription. This means that an explicit algorithm needs to be defined in order to construct coherent and consistent state. The details of this mechanism are specific to the particular document type. See Section 4.3 for information on constructing coherent information from an application/dialog-info+xml document.

3.9 Handling of Forked Requests

Since dialog state is distributed across the UA for a particular user, it is reasonable and useful for a SUBSCRIBE request for dialog state to fork, and reach multiple UA.

As a result, a forked SUBSCRIBE request for dialog state can install multiple subscriptions. Subscribers to this package MUST be prepared to install subscription state for each NOTIFY generated as a result of a single SUBSCRIBE.

3.10 Rate of Notifications

For reasons of congestion control, it is important that the rate of notifications not become excessive. As a result, it is RECOMMENDED that the server not generate notifications for a single subscriber at a rate faster than once every 1 second.

3.11 State Agents

Dialog state is ideally maintained in the user agents in which the dialog resides. Therefore, the elements that maintain the dialog are the ones best suited to handle subscriptions to it. However, in some cases, a network agent may also know the state of the dialogs held by a user. As such, state agents MAY be used with this package.

4. Dialog Information Format

Dialog information is an XML document [4] that MUST be well-formed and SHOULD be valid. Dialog information documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying dialog information documents and document fragments. The namespace URI for elements defined by this specification is a URN [5], using the namespace identifier 'ietf' defined by [6] and extended by [7]. This URN is:

urn:ietf:params:xml:ns:dialog-info

A dialog information document begins with the root element tag "dialog-info".

4.1 Structure of Dialog Information

A dialog information document starts with a dialog-info element. This element has three mandatory attributes:

version: This attribute allows the recipient of dialog information documents to properly order them. Versions start at 0, and increment by one for each new document sent to a subscriber. Versions are scoped within a subscription. Versions MUST be representable using a 32 bit integer.

state: This attribute indicates whether the document contains the full dialog information, or whether it contains only information on those dialogs which have changed since the previous document (partial).

entity: This attribute contains a URI that identifies the user whose dialog information is reported in the remainder of the document. This user is referred to as the "observed user".

The dialog-info element has a series of zero or more dialog sub-elements. Each of those represents a specific dialog.

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="0" notify-state="full"
             entity="sip:alice@example.com">
</dialog-info>
```

4.1.1 Dialog Element

The dialog element reports information on a specific dialog or "half-dialog". It has single mandatory attribute: id. The id attribute provides a single string that can be used as an identifier for this dialog or "half-dialog". This is a different identifier than the dialog ID defined in RFC 3261 [2], but related to it.

For a caller, the id is created when an INVITE request is sent. When a 1xx with a tag, or a 2xx is received, the dialog is formally created. The id remains unchanged. However, if an additional 1xx or 2xx is received, resulting in the creation of another dialog (and resulting FSM), that dialog is allocated a new id.

For a callee, the id is created when an INVITE outside of an existing

dialog is received. When a 2xx or a 1xx with a tag is sent, creating the dialog, the id remains unchanged.

The id MUST be unique amongst all dialogs at a UA.

There are a number of optional attributes which provide identification information about the dialog:

call-id: This attribute is a string which represents the call-id component of the dialog identifier. (Note that single and double quotes inside a call-id must be escaped using &quote; for " and ' for ' .)

local-tag: This attribute is a string which represents the local-tag component of the dialog identifier.

remote-tag: This attribute is a string which represents the remote-tag component of the dialog identifier. The remote tag attribute won't be present if there is only a "half-dialog", resulting from the generation of an INVITE for which no final responses or provisional responses with tags has been received.

direction: This attribute is either initiator or recipient, and indicates whether the observed user was the initiator of the dialog, or the recipient of the INVITE that created it.

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="0" state="partial"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" direction="initiator">
...
  </dialog>
</dialog-info>
```

The sub-elements of the dialog element provide additional information about the dialog. Some of these sub-elements provide more detail about the dialog itself, while the local and remote sub-elements describe characteristics of the participants involved in the dialog. The only mandatory sub-element is the state element.

4.1.2 State

The state element indicates the state of the dialog. Its value is an enumerated type describing one of the states in the FSM above. It has an optional event attribute that can be used to indicate the event which caused any transition into the terminated state, and an

optional code attribute that indicates the response code associated with any transition caused by a response to the original INVITE.
<state event="rejected" code="486">terminated</state>

4.1.3 Duration

The duration element contains the amount of time, in seconds, since the FSM was created.
<duration>145</duration>

4.1.4 Replaces

The replaces element is used to correlate a new dialog with one it replaced as a result of an invitation with a Replaces header. This element is present in the replacement dialog only (the newer dialog) and contains attributes with the call-id, local-tag, and remote-tag of the replaced dialog.
<replaces call-id="hg287s98s89" local-tag="6762h7" remote-tag="09278hsb"/>

4.1.5 Referred-By

The referred-by element is used to correlate a new dialog with a REFER [12] request which triggered it. The element is present in a dialog which was triggered by a REFER request which contained a Referred-By [11] header and contains the (optional) display name attribute and the Referred-By URI as its value.
<referred-by display="Bob">sip:bob@example.com</referred-by>

4.1.6 Local and Remote elements

The local and remote elements are sub-elements of the dialog element which contain information about the local and remote participants respectively. They both have a number of optional sub-elements which indicate the identity conveyed by the participant, the target URI, the feature-tags of the target, and the session-description of the participant.

4.1.6.1 Identity

The identity element indicates a local or remote URI, as defined in [2] as appropriate. It has an optional attribute, display, that contains the display name from the appropriate URI.

> Note that multiple identities (for example a sip: URI and a tel: URI) could be included if they all correspond to the participant. To avoid repeating identity information in each request, the subscriber can assume that the identity URIs are the same as in previous notifications if no identity elements are present in the corresponding local or remote element. If any identity elements are present in the local or remote part of a notification, the new list of identity tags completely supersedes the old list in the corresponding part.

<identity display="Anonymous">sip:anonymous@anonymous.invalid</identity>

4.1.6.2 Target

The target contains the local or remote target URI as constructed by the user agent for this dialog, as defined in RFC 3261 [2] in a "uri" attribute.

It can contain a list of Contact header parameters in param sub-elements (such as those defined in [10]. The param element contains a required pname attribute and an optional pval attribute (some parameters merely exist and have no explicit value). The param element itself has no contents. To avoid repeating Contact information in each request, the subscriber can assume that the target URI and parameters are the same as in previous notifications if no target element is present in the corresponding local or remote element. If a target element is present in the local or remote part of a notification, the new target tag and list of an parameter tags completely supersedes the old target and parameter list in the corresponding part.

```
<target uri="sip:alice@pc33.example.com">
  <param pname="isfocus"/>
  <param pname="class" pval="personal"/>
</target>
```

4.1.6.3 Session Description

The session-description element contains the session description used by the observed user for its end of the dialog. This element should generally NOT be included in the notifications, unless explicitly requested by the subscriber. It has a single attribute, type, which indicates the MIME media type of the session description. To avoid repeating session description information in each request, the subscriber can assume that the session description is the same as in previous notifications if no session description element is present in the corresponding local or remote element.

4.2 Sample Notification Body

```
<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
     xsi:schemaLocation="urn:ietf:params:xml:ns:dialog-info"
     version="1" state="full">
     <dialog id="123456">
        <state>confirmed</state>
        <duration>274</duration>
        <local>
          <identity display="Alice">sip:alice@example.com</identity>
          <target uri="sip:alice@pc33.example.com">
            <param pname="isfocus"/>
            <param pname="class" pval="personal"/>
          </target>
        </local>
        <remote>
          <identity display="Bob">sip:bob@example.org</identity>
          <target uri="sip:bobster@phone21.example.org"/>
        </remote>
     </dialog>
   </dialog-info>
```

4.3 Constructing Coherent State

The dialog information subscriber maintains a table for the list of dialogs. The table contains a row for each dialog. Each row is indexed by an ID, present in the "id" attribute of the "dialog" element. The contents of each row contain the state of that dialog as conveyed in the document. The table is also associated with a version number. The version number MUST be initialized with the value of the "version" attribute from the "dialog-info" element in the first document received. Each time a new document is received, the value of the local version number, and the "version" attribute in the new document, are compared. If the value in the new document is one higher than the local version number, the local version number is increased by one, and the document is processed. If the value in the document is more than one higher than the local version number, the local version number is set to the value in the new document, and the document is processed. If the document did not contain full state, the subscriber SHOULD generate a refresh request to trigger a full state notification. If the value in the document is less than the local version, the document is discarded without processing.

The processing of the dialog information document depends on whether it contains full or partial state. If it contains full state, indicated by the value of the "state" attribute in the "dialog-info" element, the contents of the table are flushed. They are repopulated from the document. A new row in the table is created for each "dialog" element. If the document contains partial state, as indicated by the value of the "state" attribute in the "dialog-info"

element, the document is used to update the table. For each "dialog" element in the document, the subscriber checks to see whether a row exists for that dialog. This check is done by comparing the ID in the "id" attribute of the "dialog" element with the ID associated with the row. If the dialog doesn't exist in the table, a row is added, and its state is set to the information from that "dialog" element. If the dialog does exist, its state is updated to be the information from that "dialog" element. If a row is updated or created, such that its state is now terminated, that entry MAY be removed from the table at any time.

5. Schema

The following is the schema for the application/dialog-info+xml type:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:dialog-info"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="urn:ietf:params:xml:ns:dialog-info"
  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/2001/03/xml.xsd"/>
  <xs:element name="dialog-info">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:dialog" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax"
           minOccurs="0"  maxOccurs="unbounded"/>
      </xs:sequence>
      <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:attribute name="entity" type="xs:anyURI" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="dialog">
```

```
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="tns:state" minOccurs="1" maxOccurs="1"/>
          <xs:element name="duration" type="xs:nonNegativeInteger"
            minOccurs="0" maxOccurs="1"/>
          <xs:element name="replaces" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:attribute name="call-id" type="xs:string"
                use="required"/>
              <xs:attribute name="local-tag" type="xs:string"
                use="required"/>
              <xs:attribute name="remote-tag" type="xs:string"
                use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="referred-by" type="tns:nameaddr"
            minOccurs="0" maxOccurs="1"/>
          <xs:element name="route-set" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="hop" type="xs:string" minOccurs="1"
                  maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="local" type="tns:participant"
            minOccurs="0" maxOccurs="1"/>
          <xs:element name="remote" type="tns:participant"
            minOccurs="0" maxOccurs="1"/>
          <xs:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="call-id" type="xs:string"
          use="optional"/>
        <xs:attribute name="local-tag" type="xs:string"
          use="optional"/>
        <xs:attribute name="remote-tag" type="xs:string"
          use="optional"/>
        <xs:attribute name="direction" use="optional">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="initiator"/>
              <xs:enumeration value="recipient"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
```



```
    </xs:element>
    <xs:complexType name="participant">
      <xs:sequence>
        <xs:element name="identity" type="nameaddr"
          minOccurs="0" maxOccurs="1"/>
        <xs:element name="target" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="param" minOccurs="0"
                maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="pname" type="xs:string"
                    use="required"/>
                  <xs:attribute name="pval" type="xs:string"
                    use="optional"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
          <xs:attribute name="uri" type="xs:string" use="required"/>
        </xs:element>
        <xs:element name="session-description" type="tns:sessd"
          minOccurs="0" maxOccurs="1"/>
        <xs:element name="cseq" type="xs:nonNegativeInteger"
          minOccurs="0" maxOccurs="1"/>
        <xs:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="nameaddr">
      <xs:simpleContent>
        <xs:extension base="xs:anyURI">
          <xs:attribute name="display-name" type="xs:string"
            use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="sessd">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="type" type="xs:string" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <xs:element name="state">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
```

```
            <xs:attribute name="event" use="optional">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="cancelled"/>
                  <xs:enumeration value="rejected"/>
                  <xs:enumeration value="replaced"/>
                  <xs:enumeration value="local-bye"/>
                  <xs:enumeration value="remote-bye"/>
                  <xs:enumeration value="error"/>
                  <xs:enumeration value="timeout"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="code" use="optional">
              <xs:simpleType>
                <xs:restriction base="xs:positiveInteger">
                  <xs:minInclusive value="100"/>
                  <xs:maxInclusive value="699"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

6. Examples

6.1 Basic Example

For example, if a UAC sends an INVITE that looks like, in part:

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@example.com>
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.example.com>
Content-Type: application/sdp
Content-Length: 142

[SDP not shown]
```


The XML document in a notification from Alice might look like:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="0"
             state="full"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" direction="initiator">
    <state>trying</state>
  </dialog>
</dialog-info>
```

If the following 180 response is received:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@example.com>;tag=456887766
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@host.example.com>
```

The XML document in a notification might look like:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="1"
             state="full"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" remote-tag="456887766"
          direction="initiator">
    <state>early</state>
  </dialog>
</dialog-info>
```

If it receives a second 180 with a different tag:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@example.com>;tag=hh76a
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
```

```
CSeq: 314159 INVITE
Contact: <sip:jack@host.example.com>
```

This results in the creation of a second dialog:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="2"
             state="full"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" remote-tag="456887766"
          direction="initiator">
    <state>early</state>
  </dialog>
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" remote-tag="hh76a"
          direction="initiator">
    <state>early</state>
  </dialog>
</dialog-info>
```

If a 200 OK is received on the second dialog, it moves to confirmed:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="3"
             state="partial"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" remote-tag="hh76a"
          direction="initiator">
    <state>confirmed</state>
  </dialog>
</dialog-info>
```

32 seconds later, the other early dialog terminates because no 2xx is received for it. This implies that it was successfully cancelled, and therefore the following notification is sent:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="4"
             state="partial"
             entity="sip:alice@example.com">
```



```
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" remote-tag="hh76a"
          direction="initiator">
    <state event="cancelled">terminated</state>
  </dialog>
</dialog-info>
```

6.2 Emulating a Shared-Line phone system

The following example shows how a SIP telephone user agent can provide detailed state information and also emulate a shared-line telephone system (the phone "lies" about having a dialog while it is merely offhook).

Idle:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="0" state="full"
             entity="sip:alice@example.com">
</dialog-info>
```

Seized:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="1" state="partial"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8">
    <state>trying</state>
  </dialog>
</dialog-info>
```

Dialing:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="2" state="partial"
             entity="sip:alice@example.com">
  <dialog id="as7d900as8" call-id="a84b4c76e66710"
          local-tag="1928301774" direction="initiator">
    <state>trying</state>
    <local>
      <identity display="Alice Smith">
         sip:alice@example.com
      </identity>
      <target uri="sip:alice.gruu@srv3.example.com;grid0987"/>
```

```
      </local>
      <remote>
        <identity>sip:bob@example.net</identity>
      </remote>
    </dialog>
  </dialog-info>
```

Ringing:

```
  <?xml version="1.0"?>
  <dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
               version="3" state="partial"
               entity="sip:alice@example.com">
    <dialog id="as7d900as8" call-id="a84b4c76e66710"
            local-tag="1928301774"
            remote-tag="07346y131" direction="initiator">
      <state code="180">early</state>
      <remote>
        <target uri="sip:bobster@host2.example.net"/>
      </remote>
    </dialog>
  </dialog-info>
```

Answered (by voicemail):

```
  <?xml version="1.0"?>
  <dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
               version="4" state="partial"
               entity="sip:alice@example.com">
    <dialog id="as7d900as8" call-id="a84b4c76e66710"
            local-tag="1928301774"
            remote-tag="07346y131" direction="initiator">
      <state reason="cancelled">terminated</state>
    </dialog>
    <dialog id="zxcvbnm3" call-id="a84b4c76e66710"
            local-tag="1928301774"
            remote-tag="8736347" direction="initiator">
      <state code="200">confirmed</state>
      <remote>
        <target uri="sip:bob-is-not-here@vm.example.net">
          <param pname="actor" pval="msg-taker"/>
          <param pname="automaton"/>
        </target>
      </remote>
    </dialog>
  </dialog-info>
```

Alice requests voicemail for Bob's attendant.

(Alice presses "0" in North America / "9" in Europe)
Voicemail completes a transfer with Cathy

```
  <?xml version="1.0"?>
  <dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
               version="5" state="partial"
               entity="sip:alice@example.com">
    <dialog id="zxcvbnm3" call-id="a84b4c76e66710"
            local-tag="1928301774"
            remote-tag="8736347" direction="initiator">
      <state reason="replaced">terminated</state>
    </dialog>
    <dialog id="sfhjsjk12" call-id="o34oii1"
            local-tag="8903j4"
            remote-tag="78cjkus" direction="receiver">
      <state reason="replaced">confirmed</state>
      <replaces call-id="a84b4c76e66710"
            local-tag="1928301774"
            remote-tag="8736347"/>
      <referred-by>
        sip:bob-is-not-here@vm.example.net
      </referred-by>
      <local>
        <target uri="sip:alice.gruu@srv3.example.com;grid1645"/>
      </local>
      <remote>
        <identity display="Cathy Jones">
           sip:cjones@example.net
        </identity>
        <target uri="sip:line3@host3.example.net">
          <param pname="actor" pval="attendant"/>
          <param pname="automaton" pval="false"/>
        </target>
      </remote>
    </dialog>
  </dialog-info>
```

Alice and Cathy talk, Cathy adds Alice to a local conference.

```
  <?xml version="1.0"?>
  <dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
               version="6" state="partial"
               entity="sip:alice@example.com">
    <dialog id="sfhjsjk12" call-id="o34oii1"
            local-tag="8903j4"
            remote-tag="78cjkus" direction="receiver">
      <state>confirmed</state>
      <remote>
```

```
        <target uri="sip:confid-34579@host3.example.net">
          <param pname="isfocus"/>
        </target>
      </remote>
    </dialog>
  </dialog-info>
```

Alice puts Cathy on hold

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="7" state="partial"
             entity="sip:alice@example.com">
  <dialog id="sfhjsjk12" call-id="o34oii1"
          local-tag="8903j4"
          remote-tag="78cjkus" direction="receiver">
    <state>confirmed</state>
    <local>
      <target uri="sip:alice.gruu@srv3.example.com;grid=1645">
        <param pname="+activity" pval="noninteractive"/>
      </target>
    </local>
  </dialog>
</dialog-info>
```

Cathy hangs up

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="8" state="partial"
             entity="sip:alice@example.com">
  <dialog id="sfhjsjk12" call-id="o34oii1"
          local-tag="8903j4"
          remote-tag="78cjkus" direction="receiver">
    <state reason="remote-bye">terminated</state>
  </dialog>
  <dialog id="08hjh1345">
    <state>trying</state>
  </dialog>
</dialog-info>
```

Alice hangs up:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="9" state="full"
             entity="sip:alice@example.com">
</dialog-info>
```


6.3 Minimal Dialog Information with Privacy

The following example shows the same user agent providing minimal information to maintain privacy for services like automatic callback.

Onhook:

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="0" state="full"
             entity="sip:alice@example.com">
</dialog-info>
```

Offhook: (implementation/policy choice for Alice to transition to this "state" when "seized", when Trying, when Proceeding, or when Confirmed.)

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="1" state="full"
             entity="sip:alice@example.com">
  <dialog id="1">
    <state>confirmed</state>
  </dialog>
</dialog-info>
```

Onhook: (implementation/policy choice for Alice to transition to this "state" when terminated, or when no longer "seized")

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             version="2" state="full"
             entity="sip:alice@example.com">
</dialog-info>
```

7. Security Considerations

Subscriptions to dialog state can reveal sensitive information. For this reason, Section 3.6 discusses authentication and authorization of subscriptions, and provides guidelines on sensible authorization policies. All implementations of this package MUST support the digest authentication mechanism.

Since the data in notifications is sensitive as well, end-to-end SIP encryption mechanisms using S/MIME MAY be used to protect it.

8. IANA Considerations

This document registers a new MIME type, application/dialog-info+xml and registers a new XML namespace.

8.1 application/dialog-info+xml MIME Registration

MIME media type name: application

MIME subtype name: dialog-info+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [8].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [8].

Security considerations: See Section 10 of RFC 3023 [8] and Section 7 of this specification.

Interoperability considerations: none.

Published specification: This document.

Applications which use this media type: This document type has been used to support SIP applications such as call return and auto-conference.

Additional Information:

Magic Number: None

File Extension: .dif or .xml

Macintosh file type code: "TEXT"

Personal and email address for further information: Jonathan Rosenberg, <jdrosen@jdrosen.net>

Intended usage: COMMON

Author/Change controller: The IETF.

8.2 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:dialog-info

This section registers a new XML namespace, as per the guidelines in [7].

URI: The URI for this namespace is urn:ietf:params:xml:ns:dialog-info.

Registrant Contact: IETF, SIPPING working group, <sipping@ietf.org>, Jonathan Rosenberg <jdrosen@jdrosen.net>.

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>Dialog Information Namespace</title>
</head
<body>
  <h1>Namespace for Dialog Information</h1>
  <h2>urn:ietf:params:xml:ns:dialog-info</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

8.3 Schema Registration

This specification registers a schema, as per the guidelines in in [7].

URI: please assign.

Registrant Contact: IETF, SIPPING Working Group (sipping@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).

XML: The XML can be found as the sole content of Section 5.

9. Acknowledgements

The authors would like to thank Sean Olson for his comments.

Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[4] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C REC REC-xml-20001006, October 2000.

[5] Moats, R., "URN Syntax", RFC 2141, May 1997.

[6] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[7] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[8] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

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

[10] Rosenberg, J., "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", draft-ietf-sip-callee-caps-00 (work in progress), June 2003.

[11] Sparks, R., "The SIP Referred-By Mechanism", draft-ietf-sip-referredby-01 (work in progress), February 2003.

[12] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[13] Dean, R., Biggs, B. and R. Mahy, "The Session Inititation Protocol (SIP) 'Replaces' Header", draft-ietf-sip-replaces-03 (work in progress), March 2003.

Informative References

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

[15] 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.

[16] Mahy, R., "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-mwi-02 (work in progress), March 2003.

[17] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-00 (work in progress), January 2004.

[18] Sparks, R. and A. Johnston, "Session Initiation Protocol Call Control - Transfer", draft-ietf-sipping-cc-transfer-01 (work in progress), February 2003.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027
US

EMail: schulzrinne@cs.columbia.edu
URI: http://www.cs.columbia.edu/~hgs

Rohan Mahy (editor)
Cisco Systems, Inc.
5617 Scotts Valley Dr
Scotts Valley, CA 95066
USA

EMail: rohan@cisco.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgement

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

SIPPING
Internet-Draft
Expires: December 29, 2004

K. Ono
S. Tachimoto
NTT Corporation
June 30, 2004

Requirements for End-to-Middle Security for the Session Initiation Protocol (SIP)
draft-ietf-sipping-e2m-sec-reqs-03

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 29, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

A SIP User Agent (UA) does not always trust all intermediaries in its request path to inspect its message bodies and/or headers contained in its message. The UA might want to protect the message bodies and/or headers from intermediaries except those that provide services based on its content. This situation requires a mechanism called "end-to-middle security" to secure the information passed between the UA and intermediaries, which does not interfere with end-to-end security. This document defines a set of requirements for a

mechanism to achieve end-to-middle security.

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 [1].

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [2] supports hop-by-hop security using Transport Layer Security (TLS) [3] and end-to-end security using Secure MIME (S/MIME) [4]. These security mechanisms assume that a SIP UA trusts all proxy servers along its request path to inspect the message bodies contained in the message, or a SIP UA does not trust any proxy servers to do so.

However, there is a model where trusted and partially-trusted proxy servers are mixed along a message path. The partially-trusted proxy servers are only trusted to provide SIP routing, but these proxy servers are not trusted by users to inspect its data except routing headers. A hop-by-hop confidentiality service using TLS is not suitable for this model. An end-to-end confidentiality service using S/MIME is also not suitable when the intermediaries provide services based on reading the message bodies and/or headers. This problem is described in Section 23 of [2].

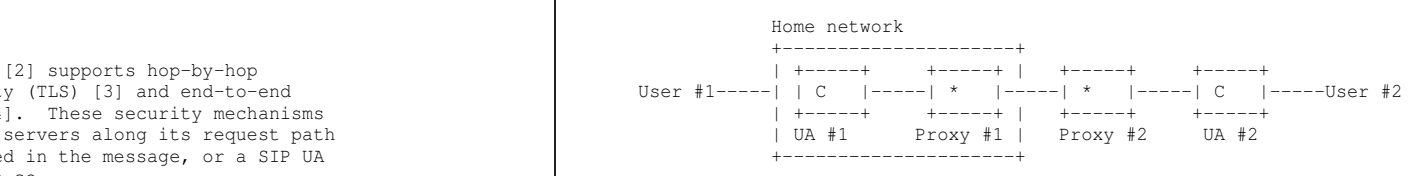
In some cases, a UA might want to protect its message bodies and/or headers from proxy servers along its request path except from those that provides services based on reading its message bodies and/or headers. Conversely, a proxy server might want to view the message bodies and/or headers to sufficiently provide these services. Such proxy servers are not always the first hop from the UA. This situation requires a security mechanism to secure message bodies and/or headers between the UA and the proxy servers, yet disclosing information to those that need it. We call this "end-to-middle security".

2. Use Cases

2.1 Examples of Scenarios

We describe here examples of scenarios in which trusted and partially-trusted proxy servers both exist in a message path. These situations demonstrate the reasons why end-to-middle security is required.

In the following example, User #1 does not know the security policies or services provided by Proxy server #1 (Proxy#1). User #1 sends a MESSAGE [5] request including S/MIME-encrypted message content for end-to-end security as shown in Figure 1, while Proxy #1 erases the encrypted data in the request or rejects the request base on its strict security policy that prohibits the forwarding of unknown data. For the MESSAGE request to correctly traverse Proxy #1, the UA will need to discover if end-to-end confidentiality will conflict with intermediary's services or security policies.

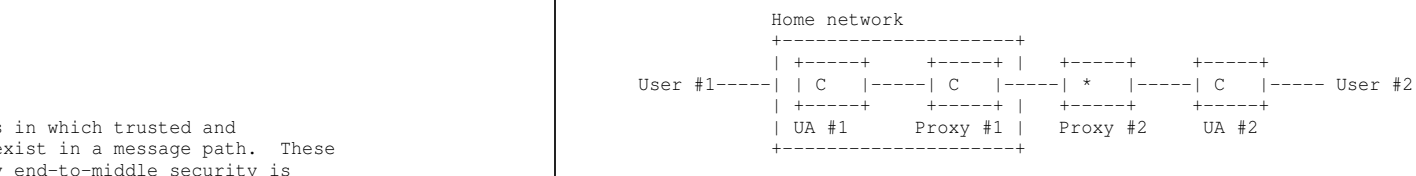


C: Content that UA #1 allows the entity to inspect
*: Content that UA #1 prevents the entity from inspecting

Figure 1: Deployment example #1

In the second example, Proxy server #1 is the home proxy server of User #1 using UA #1. User #1 communicates with User #2 through Proxy #1 and Proxy #2 as shown in Figure 2. Although User #1 already knows Proxy #1's security policy which requires the inspection of the content of the MESSAGE request, User #1 does not know whether Proxy #2 is trustworthy, and thus wants to protect the message bodies in the request. To accomplish this, UA #1 will need to be able to grant a trusted intermediary (Proxy #1) to inspect message bodies, while preserving their confidentiality from other intermediaries (Proxy #2).

Even if UA #1's request message authorizes a selected proxy server (Proxy #1) to inspect the message bodies, UA #1 is unable to authorize the same proxy server to inspect the message bodies in subsequent MESSAGE requests from UA #2.



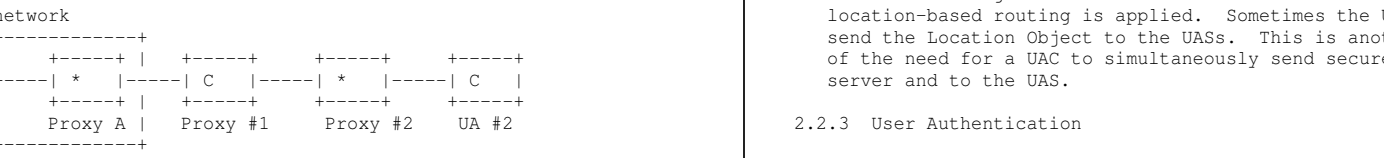
C: Content that UA #1 needs to disclose
*: Content that UA #1 needs to protect

Figure 2: Deployment example #2

In the third example, User #1 connects UA #1 to a proxy server in a visited (potentially insecure) network, e.g., a hotspot service or a roaming service. Since User #1 wants to utilize certain home network services, UA #1 connects to a home proxy server, Proxy #1. However, UA #1 must connect to Proxy #1 via the proxy server of the visited network (Proxy A), because User #1 must follow the policy of that

network. Proxy A performs access control based on the destination addresses of calls. User #1 only trusts Proxy A to route requests, not to inspect the message bodies the requests contain as shown in Figure 3. User #1 trusts Proxy #1 both to route requests and to inspect the message bodies for some purpose.

The same problems as in the second example also exist here.



C: Content that UA #1 needs to disclose
*: Content that UA #1 needs to protect

Figure 3: Deployment example #3

2.2 Service Examples

We describe here several services that require end-to-middle security.

2.2.1 Logging Services for Instant Messages

Logging Services are provided by the archiving function, which is located in the proxy server, that logs the message content exchanged between UAs. The archiving function could be located at the originator network and/or the destination network. When the content of an instant message contains private information, UACs (UA Clients) encrypt the content for the UASs (UA Servers). The archiving function needs a way to log the content in a message body in bidirectional MESSAGE requests in such a way that the data is decipherable. The archiving function also needs a way to verify the data integrity of the content before logging.

This service might be deployed in financial or health care service provider's networks, where archiving communication is required by their security policies, as well as other networks.

2.2.2 Non-emergency Call Routing Based on the Location Object

The Location Object [6] includes private information as well as routing information for appropriate proxy servers. Some proxy

servers have the capability to provide location-based routing. When UAs want to employ location-based routing in non-emergency situations, the UAs need to connect to the proxy servers with such a capability and disclose the location object contained in the message body of the INVITE request, while protecting it from other proxy servers along the request path.

The Location Object also needs to be verified for integrity before location-based routing is applied. Sometimes the UAC want to also send the Location Object to the UASs. This is another good example of the need for a UAC to simultaneously send secure data to a proxy server and to the UAS.

2.2.3 User Authentication

2.2.3.1 User Authentication using the AIBs

The Authenticated Identity Bodies (AIBs) [7] is a digitally-signed data that is used as way to identify users. Proxy servers that need to authenticate a user verify the signature. When the originator needs anonymity, the user identity in the AIB is encrypted before being signed. Proxy servers that authenticate the user need to decrypt the body in order to view the user identity in the AIB. Such proxy servers can be located at adjacent and/or non-adjacent to the UA.

The AIB could be included in all request/response messages. The proxy server needs to view it in request messages in order to authenticate users. Another proxy server sometimes needs to view it in response messages for user authentication.

2.2.3.2 User Authentication in HTTP Digest Authentication

User authentication data for HTTP digest authentication includes two types of information; potentially private information, such as a user name, and information that can be used for "replay-attacks", such as the "response" parameter that is created by a calculation using a user's password. The user authentication data can be set only in a SIP header of request messages. This information needs to be transmitted securely to servers that authenticate users, located either adjacently and/or non-adjacently to the UA.

2.2.4 Media-related Services

Firewall traversal is an example of services based on media information typically in a message body, such as the Session Description Protocol (SDP). A firewall entity that supports the SIP protocol, or a midcom [8] agent co-located with a proxy server,

controls a firewall based on media information. The SDP includes the address and port information for media streams and/or key parameters for Secure RTP (SRTP) [9]. Critical information contained in SDP requires UAs to encrypt the SDP for recipient UAs. If the SDP is encrypted for end-to-end confidentiality, the proxy server operating as a midcom agent will have no way to provide firewall traversal as it can not inspect the SDP. Therefore, there is a need for proxy server to be able to decrypt the SDP, as well as to verify the integrity of the SDP.

[Note: The validity of the use case depends on which mechanism is selected for session policies [10] by the SIPPING WG. If the session policy mechanism would require that UAs disclose media information to the policy servers using out-of-band messages, such as OPTIONS request, end-to-middle security is not required for these use cases. If the session policy mechanism employs in-band messages in order for UAs to disclose media information to the policy servers co-located with a proxy server, end-to-middle security is required. As the mechanism proposes to place subset of SDP into the header to be viewed by proxy servers, such as addresses and port numbers of media streams, these information need to be secured from entities except the policy servers.]

3. Scope of End-to-Middle Security

End-to-middle security consists of user authentication, data integrity, and data confidentiality. However, this document only describes requirements for data confidentiality and data integrity, since authentication is covered by existing mechanisms such as HTTP digest authentication [2], S/MIME Cryptographic Message Syntax (CMS) SignedData body [11], or an AIB.

As for data integrity, the CMS SignedData body can be used for verification of the data integrity by any entities. The CMS SignedData body could be used for end-to-middle security at the same time for end-to-end security.

Although a proxy server is able to verify the integrity of the data, there is no way for UAs to request a selected proxy server to verify a message with the CMS SignedData body. Therefore some new mechanisms are needed to achieve data integrity for end-to-middle security.

This document mainly discusses requirements for data confidentiality and the integrity of end-to-middle security.

4. Requirements for a Solution

We describe here requirements for a solution. The requirements are mainly applied during the phase of a dialog creation or sending a MESSAGE method.

4.1 General Requirements

The following are general requirements for end-to-middle confidentiality and integrity.

REQ-GEN-1: The solution SHOULD have little impact on the way a UA handles S/MIME-secured messages.
REQ-GEN-2: It SHOULD have no impact on proxy servers that do not provide services based on S/MIME bodies in terms of handling the existing SIP headers.
REQ-GEN-3: It SHOULD have little impact on the standardized mechanism of proxy servers in terms of handling message bodies.
REQ-GEN-4: It SHOULD allow a UA to discover security policies of proxy servers. Security policies imply what data is needed to disclose and/or verify in a message.
This requirement is necessary when the UA does not know statically which proxy servers or domains need disclosing data and/or verification.

4.2 Requirements for End-to-Middle Confidentiality

REQ-CONF-1: The solution MUST be enable an encrypted data to be shared with the recipient UA and selected proxy servers, when a UA wants.
REQ-CONF-2: It MUST NOT violate end-to-end encryption when the encrypted data does not need to be shared with any proxy servers.
REQ-CONF-3: It SHOULD allow a UA to request selected proxy servers to view specific message bodies. The request itself SHOULD be secure.
REQ-CONF-4: It SHOULD allow a UA to request that the recipient UA disclose information to the proxy server, which requesting UA is disclosing the information to. The request itself SHOULD be secure.

4.3 Requirements for End-to-Middle Integrity

REQ-INT-1: The solution SHOULD work even when the SIP end-to-end integrity service is enabled.

REQ-INT-2: It SHOULD allow a UA to request selected proxy servers to verify specific message bodies. The request itself SHOULD be secure.
REQ-INT-3: It SHOULD allow a UA to request the recipient UA to send the verification data of the same information that the requesting UA is providing to the proxy server. The request itself SHOULD be secure.

5. Security Considerations

This document describes the requirements for confidentiality and integrity between a UA and a proxy server. Although this document does not cover authentication, it is important in order to prevent attacks from malicious users and servers.

The end-to-middle security requires additional processing on message bodies, such as unpacking MIME structure, data decryption, and/or signature verification to proxy servers. Therefore the proxy servers that enable end-to-middle security are vulnerable to a Denial-of-Services attack. There is a threat model where a malicious user sends many complicated-MIME-structure messages to a proxy server, containing user authentication data obtained by eavesdropping. This attack will result in a slow down of the overall performance of these proxy servers. To prevent this attack, user authentication mechanism needs protection against replay attack. Or the user authentication always needs to be executed simultaneously with protection of data integrity. In order to prevent an attack, the following requirements should be satisfied.

- The solution MUST support mutual authentication, data confidentiality and data integrity protection between a UA and a proxy server.
- It SHOULD support protection against a replay attack for user authentication.
- It SHOULD simultaneously support user authentication and data integrity protection.

6. IANA Considerations

This document requires no additional considerations.

7. Changes

7.1 Changes from 02.txt

- Changed the text about the use case of SDP-based service in order to decrease the dependency on session policies discussion. The title was changed to "media-related service".

- Simplified the "Scope of End-to-Middle Security" section.
- Removed some of the text that described detailed information on mechanisms in the "Requirements for a Solution" section.
- Closed open issues as follows:
  * Deleted an open issue described in the "General Requirements" section, since it is no longer an issue. The issue was concerning the necessity for the proxy server to notify the UAS after receiving a response, which is not necessary, because proxy servers' security policies or services have no dependencies on the information in a response.
  * Deleted an open issue described in the "Requirements for End-to-Middle Confidentiality" section, since it is not an issue of requirements, but that of a mechanism.
- Changed the last item of the general requirements from proxy-driven to UA-driven.
- Deleted the text in the requirements that describes the relation between the requirements and the service examples.
- Added some text in the "Security Consideration" section.
- Many editorial correction.

7.2 Changes from 01.txt

- Extracted use cases from the Introduction section, and created a new section to describe the use cases in more detail. The use cases are also updated.
- Deleted a few "may" words from the "Problem with Existing Situations" section to avoid confusion with "MAY" as a key word.
- Added the relation between the requirements and the service examples.
- Deleted the redundant requirements for discovery of the targeted-middle. The requirement is described only in the "Generic Requirements", not in the "Requirements for End-to-Middle Confidentiality/Integrity".
- Changed the 4th requirement of end-to-middle confidentiality from "MUST" to "SHOULD".
- Changed the 3rd requirement of end-to-middle integrity from "MUST" to "SHOULD".
- Added some text about DoS attack prevention in the "Security Consideration" section.

7.3 Changes from 00.txt

- Reworked the subsections in Section 4 to clarify the objectives, separating end-to-middle confidentiality and integrity.

8. Acknowledgments

Thanks to Rohan Mahy and Cullen Jennings for their initial support of

this concept, and to Jon Peterson, Gonzalo Camarillo, Sean Olson, and Mark Baugher for their helpful comments.

9 References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

[4] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1992.

[5] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[6] Cuellar, J., Morris, J., Mulligan, D., Peterson, J. and J. Polk, "Geopriv Requirements", RFC 3693, February 2004.

[7] Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-03.txt (work in progress), May 2004.

[8] Srisuresh, P., Kuthan, J., Rosenberg, J., Brim, S., Molitor, A. and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[9] Baugher, M., McGrew, D., Naslund, M., Carrara, E. and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[10] Rosenberg, J., "Requirements for Session Policy for the Session Initiation Protocol (SIP)", draft-ietf-sipping-session-policy-req-01 (work in progress), February 2004.

[11] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

Authors' Addresses

Kumiko Ono
Network Service Systems Laboratories
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-shi, Tokyo 180-8585
Japan

EMail: ono.kumiko@lab.ntt.co.jp

Shinya Tachimoto
Network Service Systems Laboratories
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-shi, Tokyo 180-8585
Japan

EMail: tachimoto.shinya@lab.ntt.co.jp

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 procedures with respect to rights in RFC 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.

SIPPING
Internet-Draft
Expires: January 13, 2005

E. Burger
Brooktrout Technology, Inc.
M. Dolly
AT&T Labs

July 15, 2004

A Session Initiation Protocol (SIP) Event Package for Key Press Stimulus (KPML)

draft-ietf-sipping-kpml-04

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 13, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

The Key Press Stimulus Event Package is a component of the Applications Interaction Framework for the Session Initiation Protocol (SIP). The event package defines a Key Press Markup Language (KPML) that describes filter specifications for reporting key presses entered at a presentation-free user interface SIP User Agent (UA). The scope of this package is for collecting supplemental key presses or mid-call key presses (triggers).

This capability allows an Application Server service provider to monitor (filter) for a set of DTMF patterns at a SIP User Agent, either at an end user device or a gateway. The capability eliminates the need for hairpinning through a Media Server or duplicating all the DTMF events, when an Application Server needs to trigger mid-call service processing on DTMF digit patterns.

Conventions used in this document

RFC2119 [1] provides the interpretations for the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" found in this document.

The Application Interaction Framework [23] provides the interpretations for the terms "User Device", "SIP Application", and "User Input". This document uses the term "Application" and "Requesting Application" interchangeably with "SIP Application".

The Application Interaction Framework discusses User Device Proxies. A common instantiation of a User Device Proxy is a Public-Switched Telephone Network (PSTN) gateway. Because the normative behavior of a presentation-free user interface is identical for a presentation-free SIP User Agent and a presentation-free User Device Proxy, this document uses "User Device" for both cases.

Table of Contents

1. Introduction

This document describes the Key Press Stimulus Event Package. The Key Press Stimulus Package is a SIP Event Notification Package [5] that uses the SUBSCRIBE and NOTIFY methods of SIP. The subscription filter and notification report bodies use the Keypad Markup Language, KPML. KPML is a markup [21] that enables presentation-free user interfaces as described in the Application Interaction Framework [23].

In particular, KPML enables "dumb phones" and gateways to dumb phones to report user key-press events. Colloquially, this mechanism provides for "digit reporting" or "Dual-Tone Multi-Frequency (DTMF) reporting."

A goal of KPML is to fit in an extremely small memory and processing footprint.

The name of the markup, KPML, reflects its legacy support role. The public switched telephony network (PSTN) accomplished end-to-end signaling by transporting DTMF tones in the bearer channel. This is in-band signaling.

Voice-over-IP networks transport in-band signaling with actual DTMF waveforms or RFC2833 [12] packets. In RFC2833, the signaling application inserts RFC2833 named signal packets as well as or instead of generating tones in the media path. The receiving application gets the signal information in the media stream.

RFC2833 correlates the time the end user pressed a digit with the user's media. However, out-of-band signaling methods, as are appropriate for User Device to application signaling, do not need millisecond accuracy. On the other hand, they do need reliability, which RFC2833 does not provide.

RFC2833 tones are ideal for conveying telephone-events point-to-point in an RTP stream, as in the context of straightforward sessions like a 2-party call or simple, centrally mixed conference. However, there are other environments where additional or alternative requirements are needed. These other environments include protocol translation and complex call control.

An interested application could request notifications of every key press. However, many of the use cases for such signaling has the application interested in only one or a few keystrokes. Thus we need a mechanism for specifying to the User Device what stimulus the application would like notification of.

1.1 Protocol Translation Needs

Protocol translators between SIP and other IP protocols which use RTP, especially H.323 [18], are frequently implemented as a signaling-only entity which arranges for RTP media streams to travel directly between the final endpoints. This is an efficient arrangement in terms of limiting jitter and latency in the media, and allows the translator to support many more simultaneous sessions than if the translator terminated media as well.

Protocol translators may receive telephony-related events (especially signaled digits) via signaling. Likewise, a SIP 3pcc[10] controller, or a protocol translator which uses a traditional CTI (Computer Telephony Integration) protocol for control (ex: TAPI, TSAPI, JTAPI), may receive CTI commands to "insert" digits which may have originated from another application (for example, a desktop call control application). As the protocol translator or controller are not in the RTP path, it will want to send SIP signaled digits.

RTP implementations must be able to receive media from more than one source on the same receive port, so it would seem straightforward to send RTP to the target User Agent. This proposal has two problems however. If the target translator and SIP User Agent are separated by a firewall, then it is likely that this traffic from a different IP address will be discarded.

It is also unlikely that most low-end RTP implementations (IP phones, and software User Agents) will render this additional media correctly. What is more problematic is that there is no mechanism to determine if a SIP User Agent can properly insert telephony events received in an RTP stream separate from their other audio media.

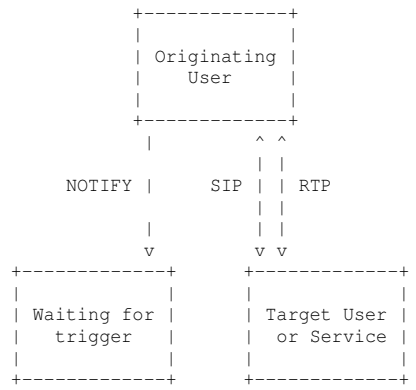
This issue is particularly apparent for H.323-SIP interworking scenarios where the H.323 network signals digits in the signaling plane using H.245 [19]. Ideally, a protocol translator should be able to signal the H.323 digits in the SIP network in the signaling plane, as well.

1.2 Complex Call Control

Some applications are interested in the telephony signals represented by telephony tones, but do not desire to be a party to the speech portion of the audio media. This document addresses the transport requirements of these signals in this context. Synchronizing speech is a non-issue in these topologies, as there is no audio media with which to synchronize; and SIP provides its own reliability mechanism to prevent loss.

For example, in some application scenarios, a user contacts an application, places a new call in the context of the application (an "outcall"), and returns to the application after the new call is finished. Examples of such scenarios include: Calling card systems, Voicemail or Messaging systems which allows outgoing calls, and Voice Browsers or Voice Portals which allow outgoing calls.

All of these applications require a way for the user to get back to the application if something has gone wrong with the outgoing call (ex: wrong number), or if the user changes his or her mind. If the originating user is using a TDM telephone, or a simple IP endpoint, the application will typically expect a sequence of signaled digits (ex: a pound or hash (#) of long duration, three stars (*) in a row, etc.)



Below are several possible SIP topologies that would enable this type of behavior. Most of these approaches fall into two categories: the application could receive DTMF media corresponding to the signaled digits, or it could receive the signaled digits using SIP.

Below are three approaches to encoding this information as media. None of these approaches are very attractive.

- The application could relay all the media itself. This wastes network resources and is inefficient for the application.
- The application could setup a conference and INVITE itself to the conference. This method requires setting up a complex set of call legs and wastes network and conferencing resources. It also requires that the application verify that the tone media originated exclusively from desired source, which may be impossible.
- The application could request "forked-media" (multi-unicast) from RFC3264 [13] of just the RFC2833 media. While the best media-related proposal, this method requires rather complex functionality in the "forking" UAs; requires 3pcc, and is problematic for firewalls because of the complexity of the SDP session description from RFC2327 [10]]. Also, experience at interoperability tests shows that most current SDP implementations are much less robust than their SIP counterparts.

2. Key Press Stimulus Operation

2.1 Model

The Key Press Stimulus reporting model is that key presses, or detected digits, are events at the User Device. The subscription installs an event filter. That event filter specifies the User Input strings, for which, if matched, causes the User Device to send a notification.

There are three usage models for the event package. Functionally, they are equivalent. However, it is useful to understand the use cases.

The principal model is that of a third-party application that is interested in the User Input. Figure 2 shows an established SIP dialog between the User Device and a SIP UA. The Requesting Application addresses the particular media stream (From RTP [9] port B to RTP port Y) by referencing the dialog identifier referring to the dialog between SIP ports A and X.

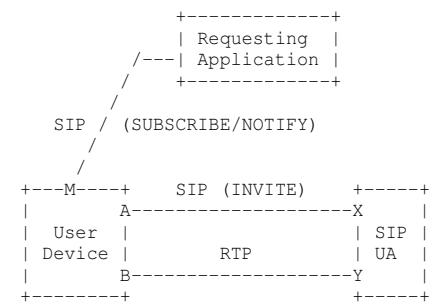


Figure 2: Third-Party Model

The second scenario is when the Application is co-resident with the remote SIP User Agent (UA). Note the application creates a separate, SUBSCRIBE-initiated dialog, as diagrammed in Figure 3. This scenario represents, for example, a toll by-pass situation where the User Device is an ingress gateway and the SIP UA is an egress gateway.

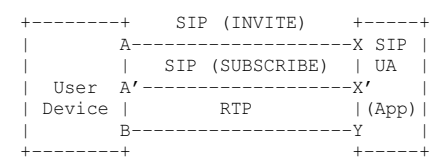


Figure 3: Endpoint Model

The third model is that of a User Device Proxy, as described by App Interaction [23]. The User Device in Figure 4 is a media relay in the terminology of RFC1889 [9]. However, in addition to the RTP forwarding capability of a RFC1889 media relay, the media proxy can also do light media processing, such as tone detection, tone transcoding (tones to RFC2833 [12]), and so on.

The Requesting Application uses dialog identifiers to identify the stream to monitor. The default is to monitor the media entering the User Device. For example, if the Requesting Application in Figure 4 refers to the dialog represented by SIP ports V-C, then the media coming from SIP UAa RTP port W gets monitored. Likewise, the dialog represented by A-X directs the User Device to monitor the media coming from SIP UAb RTP Port Y.

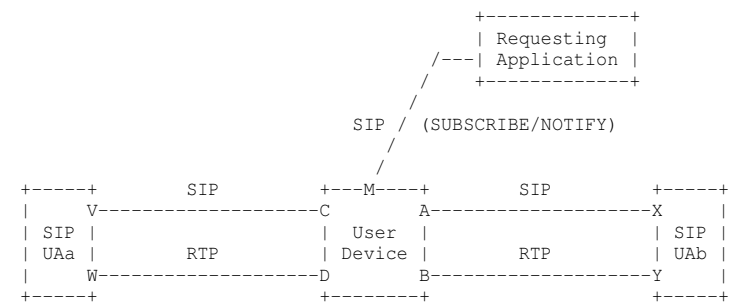


Figure 4: Media Proxy Model

2.2 Stream to Monitor

The default media stream to monitor is the stream represented by the first m= line of the SDP referenced by the dialog with the local tag of the SIP dialog at the monitoring User Device. The User Device MAY offer other streams for monitoring. One possibility is the remote stream representing the state of the device at the other end of the SIP dialog.

The User Device MUST be able to report on local User Input. In the case where the User Device is a gateway, that is, it is a User Device Proxy, local User Input is the media stream that emanates from the User Device.

If the requesting application wishes to monitor multiple streams at a given User Device, the application MUST establish multiple subscriptions, one for each stream.

2.3 Operation

The Key Press Stimulus Event Package uses explicit subscription notification requests, using the SUBSCRIBE/NOTIFY [5] mechanism.

The User Device MUST return a Contact URI that has GRUU [26] properties in the Contact header of a SIP INVITE, 1xx, or 2xx response.

Following the semantics of SUBSCRIBE, if the User Device receives a second subscription on the same dialog, including id, if present, the User Device MUST terminate the existing KPML subscription and replace

it with the new subscription.

An Application MAY register multiple User Input patterns in a single KPML subscription.

If the User Device supports multiple, simultaneous KPML subscriptions, the Application installs the subscriptions either in a new SUBSCRIBE-initiated dialog or on an existing SUBSCRIBE-initiated dialog with a new event id tag.

If the User Device does not support multiple, simultaneous KPML subscriptions, the User Device MUST respond with a KPML status code.

A KPML subscription can be persistent or one-shot. Persistent requests are active until either the dialog terminates, including normal subscription expiration, the Application replaces them, the Application deletes them by sending a null document on the dialog, or the Application deletes the subscription by sending a SUBCRIBE with an expires of zero (0).

Standard SUBSCRIBE processing dictates the User Device sends a NOTIFY response if it receives a SUBSCRIBE with an expires of zero.

One-shot requests terminate themselves once a match occurs. The "persist" KPML element specifies whether the subscription remains active for the duration specified in the SUBSCRIBE message or if it automatically terminates after a pattern matches.

KPML subscriptions route to the User Device using standard SIP request routing. A KPML subscription identifies the media stream by referencing its dialog identifiers.

Notifications are KPML documents. If the User Device matched a digit map, the response indicates the User Input detected and whether the User Device suppressed User Input. If the User Device had an error, such as a timeout, it will indicate that instead.

3. Event Package Operation

The following sub-sections are the formal specification of the KPML SIP-specific event notification package.

3.1 Event Package Name

The name for the Key Press Stimulus Event Package is "kpml".

3.2 Event Package Parameters

SIP identifies dialogs by their dialog identifier. The dialog identifier is the remote-tag, local-tag, and Call-ID entities.

To identify a specific dialog, all three of these parameters MUST be present. Usually, the local-tag is the To: entity with the To tag, the remote-tag is the From: entity including tag, and the call-id matches the Call-ID.

There may be ambiguity in specifying only the SIP dialog to monitor. The dialog may specify multiple SDP streams that could carry key press events. For example, a dialog may have multiple audio streams. Wherever possible, the User Device MAY apply local policy to disambiguate which stream or streams to monitor. In order to have an extensible mechanism for identifying streams, the mechanism for specifying streams is as an element content to the <stream> tag. The only content defined today is the <stream>reverse</stream> tag.

For most situations, such as a monaural point-to-point call with a single codec, the stream to monitor is obvious. In such situations the Application need not specify which stream to monitor.

The BNF for these parameters is as follows. The definitions of callid, token, EQUAL, and DQUOTE are from RFC3261 [4].

```
call-id   =  "call-id" EQUAL DQUOTE callid DQUOTE
from-tag  =  "from-tag" EQUAL token
to-tag    =  "to-tag" EQUAL token
```

The call-id parameter is a quoted string. This is because the BNF for word (which is used by callid) allows for characters not allowed within token. One usually just copies these elements from the Call-Id, to, and from fields of the SIP INVITE.

One can use any method of determining the dialog identifier. One method available, particularly for third-party applications, is the SIP Dialog Package [27].

Figure 6 Shows a subscription that identifies the dialog labled with the To Tag "jfg777666bc", From Tag "002993bbcdc", and Call ID "12@example.com". Note the pretty-printing. The parameters to the kpml event go on the same line as the event specification.

```
SUBSCRIBE sip:a-real-gruu@ud.example.net SIP/2.0
From: <sip:app@example.com>;tag=023948asdcn
To: <sip:a-real-gruu@ud.example.net>;tag=jfq498489qb
Call-Id: 349f8jasdvn@example.com
...
Event: kpml
       ; to-tag=jfg777666bc
       ; from-tag=002993bbcdc
       ; call-id=12@example.com
...
```

Figure 6: Identifying a Dialog

3.3 SUBSCRIBE Bodies

KPML specifies key press event notification filters. The MIME type for KPML requests is application/kpml-request+xml.

The KPML request document MUST be well-formed and SHOULD be valid. KPML documents MUST conform to XML 1.0 [21] and MUST use UTF-8 encoding.

Because of the potentially sensitive nature of the information reported by KPML, subscribers SHOULD use sips: and SHOULD consider the use of S/MIME on the content.

Subscribers MUST be prepared for the notifier to insist on authentication at a minimum and to expect encryption on the documents.

3.4 Subscription Duration

The "persist" attribute to the <pattern> tag in the KPML subscription body affects the lifetime of the subscription.

If the persist attribute is "one-shot", then once there is a match (or no match is possible), the subscription ends after the User Device notifies the Application.

If the persist attribute is "persist" or "single-notify", then the subscription ends when the Application explicitly ends it or the User Device terminates the subscription.

The subscription lifetime MUST NOT be longer than the negotiated expires time, per RFC3265 [5].

The subscription lifetime should be longer than the expected call

time. The default subscription lifetime (Expires value) MUST be 7200 seconds.

Subscribers MUST be able to handle the User Device returning an Expires value smaller than the requested value. Per RFC3265 [5], the subscription duration is the value returned by the User Device in the 200 OK Expires entity.

3.5 NOTIFY Bodies

KPML specifies the key press notification report format. The MIME type for KPML reports is application/kpml-response+xml. The default MIME type for the kpml event package is application/ kpml-response+xml.

If the requestor is not using a secure transport protocol such as TLS (e.g., by using a sips: URI), the User Device SHOULD use S/MIME to protect the user information in responses.

3.6 Notifier Processing of SUBSCRIBE Requests

The user information transported by KPML is potentially sensitive. For example, it could include calling card or credit card numbers. Thus the first action of the User Device (notifier) SHOULD be to authenticate the requesting party.

User Devices MUST support digest authentication at a minimum.

User Devices MUST support the sips: scheme and TLS.

Upon authenticating the requesting party, the User Device determines if the requesting party has authorization to monitor the user's key presses. Determining authorization policies and procedures is beyond the scope of this specification.

> NOTE: While it would be good to require both authorization and user notification for KPML, some uses, such as lawful intercept pen registers, have very strict authorization requirements yet have a requirement of no user notification. Conversely, pre-paid applications running on a private network may have no authorization requirements and already have implicit user acceptance of key press monitoring. Thus we cannot give any normative rules here.

After authorizing the request (RECOMMENDED), the User Device checks to see if the request is to terminate a subscription. If the request will terminate the subscription, the User Device does the appropriate processing, including the procedures described in Section 3.7.4.

If the request has no KPML body, then any KPML document running on that dialog, and addressed by the event id, if present, immediately terminates. This is a mechanism for unloading a KPML document while keeping the SUBSCRIBE-initiated dialog active. This can be important for secure sessions that have high costs for session establishment, such as TLS. The User Device follows the procedures described in Section 3.7.1.

If the dialog referenced by the kpml subscription does not exist, the User Device follows the procedures in Section 3.7.5 Note the User Device MUST issue a 200 OK before issuing the NOTIFY, as the SUBSCRIBE itself is well-formed.

If the request has a KPML body, the User Device parses the KPML document. The User Device SHOULD validate the XML document against the schema presented in Section 6.2. If the document is not valid, the User Device performs the procedures described in Section 3.7.6. If there is a loaded KPML document on the dialog (and given event id, if present), the User Device unloads the document.

In addition, if there is a loaded KPML document on the dialog (with the given event id, if present), the end device unloads the document.

3.7 Notifier Generation of NOTIFY Requests

3.7.1 SIP Protocol-Generated

The User Device (notifier in SUBSCRIBE/NOTIFY parlance) generates NOTIFY requests based on the requirements of RFC3265 [5]. Specifically, unless a SUBSCRIBE request is not valid, all SUBSCRIBE requests will result in an immediate NOTIFY.

The KPML payload distinguishes between a NOTIFY that RFC3265 mandates and a NOTIFY informing of key presses. If there is no User Input buffered at the time of the SUBSCRIBE (see Section 4.1 below) or the buffered User Input does not match the new KPML document, then the immediate NOTIFY MUST NOT contain a KPML body. If User Device has User Input buffered that result in a match using the new KPML document, then the NOTIFY MUST return the appropriate KPML document.

All subscriptions MUST be authenticated, particularly those that match on buffered input.

3.7.2 Match

During the subscription lifetime, the User Device may detect a key press stimulus that triggers a KPML event. In this case, the User Device (notifier) MUST return the appropriate KPML document.

3.7.3 Inter-Digit Timeout No Match

Once a user starts to enter stimulus, it is highly likely they will enter all of the key presses of interest within a specific time period. There is a temporal locality of reference for key presses. It is possible for users to accidentally press a key, however. Moreover, users may start pressing a key and then be lost as to what to do next. For applications to handle this situation, KPML allows applications to request notification if the user starts to enter stimulus but then stops before a match.

Once the User Device detects a key press that matches the first character of a digit map, the User Device starts the interdigit timer specified in the <pattern> tag. Every subsequent key press detected restarts the interdigit timer. If the interdigit timer expires, the User Device generates a KPML report with the KPML status code 423, Timer Expired. The report also includes the User Input collected up to the time the timer expired. This could be the null string. After sending the NOTIFY, the User Device will resume quarantining additional detected User Input.

Applications may have different requirements for the interdigit timer. For example, applications targeted to user populations that tend to key in information slowly may require longer interdigit timers. The specification of the interdigit timer is in milliseconds. The default value is 4000, for 4 seconds. A value of zero indicates disabling the interdigit timer. The User Device MUST round up the requested interdigit timer to the nearest time increment it is capable of detecting.

3.7.4 Dialog Terminated

It is possible for a dialog to terminate during key press collection. The cases enumerated here are explicit subscription termination, automatic subscription termination, and underlying (INVITE-initiated) dialog termination.

If a SUBSCRIBE request has an expires of zero (explicit SUBSCRIBE termination), includes a KPML document, and there is buffered User Input, then the User Device attempts to process the buffered digits against the document. If there is a match, the User Device MUST generate the appropriate KPML report with the KPML status code of 200. The SIP NOTIFY body terminates the subscription by setting the subscription state to "terminated" and a reason of "timeout".

If the SUBSCRIBE request has an expires of zero and no KPML body or the expires timer on the SUBSCRIBE-initiated dialog fires at the User Device (notifier), then the User Device MUST issue a KPML report with

the KPML status code 487, Subscription Expired. The report also includes the User Input collected up to the time the expires timer expired or when the subscription with expires equal to zero was processed. This could be the null string.

Per the mechanisms of RFC3265 [5], the User Device MUST terminate the SIP SUBSCRIBE dialog. The User Device does this via the SIP NOTIFY body transporting the final report described in the preceding paragraph. In particular, the subscription state will be "terminated" and a reason of "timeout".

Terminating the subscription when a dialog terminates ensures reauthorization (if necessary) for attaching to subsequent subscriptions.

3.7.5 Dialog Not Present

If a SUBSCRIBE request references a dialog that is not present at the User Device, the User Device MUST generate a KPML report with the KPML status code 481, Dialog Not Found. The User Device terminates the subscription by setting the subscription state to "terminated".

3.7.6 Bad Document

If the KPML document is not valid, the User Device generates a KPML report with the KPML status code 501, Bad Document. The User Device terminates the subscription by setting the subscription state to "terminated".

If the document is valid but the User Device does not support a namespace in the document, the User Device MUST respond with a KPML status code 502, Namespace Not Supported.

3.7.7 One-Shot vs. Persistent Requests

There are two types of subscriptions: one-shot and persistent. Persistent subscriptions have two sub-types: continuous notify and single-notify.

One-shot subscriptions terminate after a pattern match and report. If the User Device detects a key press stimulus that triggers a one-shot KPML event, then the User Device (notifier) MUST set the "Subscription-State" in the NOTIFY message to "terminated". At this point the User Device MUST consider the subscription destroyed.

Persistent subscriptions remain active at the User Device, even after a match. For continuous notify persistent subscriptions, the User Device will emit a notification whenever the User Input matches a

pattern. For single-notify persistent subscriptions, the User Device will emit a notification at the first match, but will not emit further notifications until the Application issues a new document on the subscription dialog.

> NOTE: The single-notify persistent subscription enables lock-step (race-free) quarantining of User Input between different digit maps.

3.8 Subscriber Processing of NOTIFY Requests

3.8.1 No KPML Body

If there is no KPML body, it means the SUBSCRIBE was successful. This establishes the dialog if there is no buffered User Input to report.

3.8.2 KPML Body

If there is a KPML document, and the KPML status code is 200, then a match occurred.

If there is a KPML document, and the KPML status code is 4xx, then an error occurred with User Input collection. The most likely cause is a timeout condition.

If there is a KPML document, and the KPML status code is 5xx, then an error occurred with the subscription. See Section 7 for more on the meaning of KPML status codes.

The subscriber MUST be mindful of the subscription state. The User Device may terminate the subscription at any time.

3.9 Handling of Forked Requests

The SUBSCRIBE behavior described in Section 3.6 ensures that it is only possible to have a subscription where there is an active (e.g., voice) dialog. Thus the case of multiple subscription installation cannot occur.

3.10 Rate of Notifications

The User Device MUST NOT generate messages faster than 25 messages per second, or one message every 40 milliseconds. This is the minimum time period for MF digit spills. Even 30-millisecond DTMF, as one sometimes finds in Japan, has a 20-millisecond off time, resulting in a 50-millisecond interdigit time. This document strongly RECOMMENDS AGAINST using KPML for digit-by-digit messaging,

such as would be the case if the only <regex> is "x".

The sustained rate of notification shall be no more than 100 Notifies per minute.

The User Device MUST reliably deliver notifications. Because there is no meaningful metric for throttling requests, the User Device SHOULD send NOTIFY messages over a congestion-controlled transport, such as TCP or SCTP.

User Devices MUST at a minimum implement SIP over TCP.

3.11 State Agents

Not applicable.

4. Message Format - KPML

The Key Press Markup Language (KPML) consists of two schemas, the kpml-request and kpml-response.

4.1 KPML Request

A KPML request document contains a <pattern> element with a series of <regex> tags. The <regex> element specifies a pattern for the User Device to report on. Section 5 describes the DRegex, or digit regular expression, language.

4.1.1 User Input Buffer Behavior

User Devices MUST NOT buffer USER input prior to an authenticated subscription, unless the INVITE establishing the dialog includes "Require: kpml".

NOTE: This is a first stab at some sort of programmatic method of starting buffering without buffering everything all the time.

User Devices MUST buffer User Input upon receipt of an authenticated and accepted subscription. Subsequent KPML documents apply their patterns against the buffered User Input. Some applications use modal interfaces where the first few key presses determine what the following key presses mean. For a novice user, the application may play a prompt describing what mode the application is in. However, "power users" often barge through the prompt.

KPML provides a <flush> tag in the <pattern> element. The default is not to flush User Input. Flushing User Input has the effect of ignoring key presses entered before the installation of the KPML subscription. To flush User Input, include the tag

<flush>yes</flush>
in the KPML subscription document. Note that this directive affects only the current subscription dialog/id combination.

Lock-step processing of User Input is where the User Device issues a notification, the Application processes the notification while the User Device buffers additional User Input, the Application requests more User Input, and only then does the User Device notify the Application based on the collected User Input. To direct the User Device to operate in lock-step mode, set the <pattern> attribute persist="single-notify".

The User Device MUST be able to process <flush>no</flush>. This directive is effectively a no-op.

Other string values for <flush> may be defined in the future. If the User Device receives a string it does not understand, it MUST treat the string as a no-op.

If the user presses a key that cannot match any pattern within a <regex> tag, the User Device MUST discard all buffered key presses up to and including the current key press from consideration against the current or future KPML documents on a given dialog. However, as described above, once there is a match, the User Device buffers any key presses the user entered subsequent to the match.

NOTE: This behavior allows for applications to only receive User Input that interest them. For example, a pre-paid application only wishes to monitor for a long pound. If the user enters other stimulus, presumably for other applications, the pre-paid application does not want notification of that User Input. This feature is fundamentally different than the behavior of TDM-based equipment where every application receives every key press.

To limit reports to only complete matches, set the "nopartial" attribute to the <pattern> tag to "true". In this case, the User Device attempts to match a rolling window over the collected User input.

KPML subscriptions are independent. Thus it is not possible for the current document to know if a following document will enable barging or want User Input flushed. Therefore, the User Device MUST buffer all User Input, subject to the forced_flush caveat described below.

On a given SUBSCRIBE dialog with a given id, the User Device MUST buffer all User Input detected between the time of the report and the receipt of the next document, if any. If the next document indicates a buffer flush, then the interpreter MUST flush all collected User

Input from consideration from KPML documents received on that dialog with the given event id. If the next document does not indicate flushing the buffered User Input, then the interpreter MUST apply the collected User Input (if possible) against the digit maps presented by the script's <regex> tags. If there is a match, the interpreter MUST follow the procedures in Section 3.7.2. If there is no match, the interpreter MUST flush all of the collected User Input.

Given the potential for needing an infinite buffer for User Input, the User Device MAY discard the oldest User Input from the buffer. If the User Device discards digits, when the User Device issues a KPML notification, it MUST set the forced_flush attribute of the <response> tag to "true". For future use, the Application MUST consider any non-null value, other than "false" that it does not understand, to be the same as "true".

NOTE: The requirement to buffer all User Input for the entire length of the session is not really onerous under normal operation. For example, if one has a gateway with 8,000 sessions, and the gateway buffers 50 key presses on each session, the requirement is only 400,000 bytes, assuming one byte per key press.

Unless there is a suppress indicator in the digit map, it is not possible to know if the User Input is for local KPML processing or for other recipients of the media stream. Thus, in the absence of a suppression indicator, the User Device transmits the User Input to the far end in real time, using either RFC2833, generating the appropriate tones, or both.

The section Digit Suppression (Section 4.1.3) describes the operation of the suppress indicator.

4.1.2 Pattern Matching

4.1.2.1 Inter-Digit Timing

The pattern matching logic works as follows. KPML User Devices MUST follow the logic presented in this section so that different implementations will perform deterministically on the same KPML document given the same User Input.

The pattern match algorithm matches the longest regular expression. This is the same mode as H.248.1 [16] and not the mode presented by MGCP [15]. The pattern match algorithm choice has an impact on determining when a pattern matches. Consider the following KPML document.


```
<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern>
    <regex>0</regex>
    <regex>011</regex>
  </pattern>
</kpml-request>
```

Figure 8: Greedy Matching

In Figure 8, if we were to match on the first found pattern, the string "011" would never match. This happens because the "0" rule would match first.

While this behavior is what most applications desire, it does come at a cost. Consider the following KPML document snippet.

```
  <regex>x{7}</regex>
  <regex>x{10}</regex>
```

Figure 9: Timeout Matching

Figure 9 is a typical North American dial plan. From an application perspective, users expect a seven-digit number to respond quickly, not waiting the typical inter-digit critical timer (usually four seconds). Conversely, the User does not want the system to cut off their ten-digit number at seven digits because they did not enter the number fast enough.

One approach to this problem is to have an explicit dial string terminator. Typically, it is the pound key (#). Now, consider the following snippet.

```
<regex>x{7}#</regex>
<regex>x{10}#</regex>
```

Figure 10: Timeout Matching with Enter

The problem with the approach in Figure 10 is that the digit collector will still look for a digit after the "#" in the seven-digit case. Worse yet, the "#" will appear in the returned dial string.

The approach used in KPML is to have an explicit "Enter Key", as

shown in the following snippet.

```
<pattern enterkey="#">
  <regex>x{7}</regex>
  <regex>x{10}</regex>
</pattern>
```

Figure 11: Timeout Matching with Enter Key

In Figure 11, the enterkey attribute to the <pattern> tag specifies a string that terminates a pattern. In this situation, if the user enters seven digits followed by the "#" key, the pattern matches (or fails) immediately. KPML indicates a terminated nomatch with a KPML status code 402.

NOTE: The enterkey is a string. The enterkey can be a sequence of key presses.

To address the various key press collection scenarios, we define three timers. The timers are the critical timer (criticaltimer), the inter-digit timer (interdigittimer), and the extra digit timer (extradigittimer). The critical timer is the time to wait for another digit if the collected digits can match a pattern. The extra timer is the time to wait after the longest match has occurred (presumably for the Enter key). The inter-digit timer is the time to wait between digits in all other cases. Note there is no start timer, as that concept does not apply in the KPML context.

The User Device MAY support an inter-digit timeout value. This is the amount of time the User Device will wait for User Input before returning a timeout error result on a partially matched pattern. The application can specify the inter-digit timeout as an integer number of milliseconds by using the "interdigittimer" attribute to the <pattern> tag. The default is 4000 milliseconds. If the User Device does not support the specification of an inter-digit timeout, the User Device MUST silently ignore the specification. If the User Device supports the specification of an inter-digit timeout, but not to the granularity specified by the value presented, the User Device MUST round up the requested value to the closest value it can support.

The User Device MAY support an extra-digit timeout value. This is the amount of time the User Device will wait for another key press when it already has a matched <regex>. The application can specify the extra-digit timeout as an integer number of milliseconds by using the "extradigittimer" attribute to the <pattern> tag. The default is 500 milliseconds.

The User Device MAY support a critical-digit timeout value. This is

the amount of time the User Device will wait for another key press when it already has a matched <regex> but there is another, longer <regex> that may also match the pattern. The application can specify the critical-digit timeout as an integer number of milliseconds by using the "criticaldigittimer" attribute to the <pattern> tag. The default is 1000 milliseconds.

4.1.2.2 Intra-Digit Timing

Some patterns look for long duration key presses. For example, some applications look for long "#" or long "*".

KPML uses the "L" modifier to <regex> characters to indicate long key presses. The following KPML document looks for a long pound of at least 3 seconds.

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern long="3000">
    <regex>L#</regex>
  </pattern>
</kpml-request>
```

The request can specify what constitutes "long" by setting the long attribute to the <pattern>. This attribute is an integer representing the number of milliseconds. If the user presses a key for longer than "long" milliseconds, the Long modifier is true. The default length of the long attribute is 2500 milliseconds.

Some User Devices are unable to present long key presses. An example is an old private branch exchange (PBX) phone set that emits fixed-length tones when the user presses a key. To address this issue, the User Device MAY interpret a success of a single key press to be equivalent to a long key press of the same key. The Application indicates it wants this behavior by setting the "longrepeat" attribute tot he <pattern> to "true".

4.1.3 Digit Suppression

Under basic operation, a KPML User Device will transmit in-band tones (RFC2833 [12] or actual tone) in parallel with User Input reporting.

NOTE: If KPML did not have this behavior, then a User Device executing KPML could easily break called applications. For

example, take a personal assistant that uses "*9" for attention. If the user presses the "*" key, KPML will hold the digit, looking for the "9". What if the user just enters a "*" key, possibly because they accessed an IVR system that looks for "*"? In this case, the "*" would get held by the User Device, because it is looking for the "*9" pattern. The user would probably press the "*" key again, hoping that the called IVR system just did not hear the key press. At that point, the User Device would send both "*" entries, as "**" does not match "*9". However, that would not have the effect the user intended when they pressed "*".

On the other hand, there are situations where passing through tones in-band is not desirable. Such situations include call centers that use in-band tone spills to effect a transfer.

For those situations, KPML adds a suppression tag, "pre", to the <regex> tag. There MUST NOT be more than one <pre> in any given <regex>.

If there is only a single <pattern> and a single <regex>, suppression processing is straightforward. The end-point passes User Input until the stream matches the regular expression <pre>. At that point, the User Device will continue collecting User Input, but will suppress the generation or pass-through of any in-band User Input.

If the User Device suppressed stimulus, it MUST indicate this by including the attribute "suppressed" with a value of "true" in the notification.

Clearly, if the User Device is processing the KPML document against buffered User Input, it is too late to suppress the transmission of the User Input, as the User Device has long sent the stimulus. This is a situation where there is a <pre> specification, but the "suppressed" attribute will not be "true" in the notification. If there is a <pre> tag that the User Device matched and the User Device is unable to suppress the User Input, it MUST set the "suppressed" attribute to "false".

A KPML User Device MAY perform suppression. If it is not capable of suppression, it ignores the suppression attribute. It MUST set the "suppressed" attribute to "false". In this case, the pattern to match is the concatenated pattern of pre+value.

At some point in time, the User Device will collect enough User Input to the point it hits a <pre> pattern. The interdigittimer attribute indicates how long to wait once the user enters stimulus before reporting a time-out error. If the interdigittimer expires, the User Device MUST issue a time-out report, transmit the suppressed User

Input on the media stream, and stop suppression.

Once the User Device detects a match and it sends a NOTIFY request to report the User Input, the User Device MUST stop suppression. Clearly, if subsequent User Input matches another <pre> expression, then the User Device MUST start suppression.

After suppression begins, it may become clear that a match will not occur. For example, take the expression
<regex><pre>*8</pre>xxx[2-9]xxxxxx</regex>
At the point the User Device receives "*8", it will stop forwarding stimulus. Let us say that the next three digits are "408". If the next digit is a zero or one, the pattern will not match.

NOTE: It is critically important for the User Device to have a sensible inter-digit timer. This is because an errant dot (".") may suppress digit sending forever. See Section 4.1 for setting the inter-digit timer.

Applications should be very careful to indicate suppression only when they are fairly sure the user will enter a digit string that will match the regular expression. In addition, applications should deal with situations such as no-match or time-out. This is because the User Device will hold digits, which will have obvious user interface issues in the case of a failure.

4.1.4 One-Shot and Persistent Triggers

The KPML document specifies if the patterns are to be persistent by setting the "persist" attribute to the <pattern> tag to "persist" or "single-notify". Any other value, including "one-shot", indicates the request is a one-shot subscription. If the User Device does not support persistent subscriptions, it returns a KPML document with the KPML status code set to 531. If there are digits in the buffer and the digits match an expression in the KPML document, the User Device prepares the appropriate KPML document.

Note the values of the persistent attribute are case sensitive.

4.1.5 Multiple Patterns

Some User Devices may support multiple regular expressions in a given pattern request. In this situation, the application may wish to know which pattern triggered the event.

KPML provides a "tag" attribute to the <regex> tag. The "tag" is an opaque string that the User Device sends back in the notification report upon a match in the digit map. In the case of multiple

matches, the User Device MUST chose the longest match in the KPML document. If multiple matches match the same length, the User Device MUST chose the first expression listed in the subscription KPML document based on KPML document order.

If the User Device does not support multiple regular expressions in a pattern request, the User Device MUST return a KPML document with the KPML status code set to 532.

4.1.6 Monitoring Direction

By default, the User Device monitors key presses emanating from the User Device. Given a dialog identifier of Call-ID, local-tag, and remote-tag, the User Device monitors the key presses associated with the local-tag.

In the media proxy case, and potentially other cases, there is a need to monitor the key presses arriving from the remote user agent. The optional <stream> element to the <request> tag specifies which stream to monitor. The only legal value is "reverse", which means to monitor the stream associated with the remote-tag. The User Device MUST ignore other values.

NOTE: The reason this is a tag is so individual stream selection, if needed, can be addressed in a backwards-compatible way.

NOTE: Further specification of the stream to monitor is the subject of future standardization. The current thoughts revolve around negotiating MIME parameters that describe namespaces declaring the filters specification of the stream.

4.1.7 Multiple, Simultaneous Subscriptions

Some User Devices may support multiple key press event notification subscriptions at the same time. In this situation, the User Device honors each subscription individually and independently.

A SIP user agent may request multiple subscriptions on the same SUBSCRIBE dialog, using the id parameter to the kpml event request.

One or more SIP user agents may request independent subscriptions on different SIP dialogs. Section 3.2 describes the dialog addressing mechanism in detail.

If the User Device does not support multiple, simultaneous subscriptions, the User Device MUST return a KPML document with the KPML status code set to 533 on the dialog that requested the second subscription. The User Device MUST NOT modify the state of the first subscription on account of the second subscription attempt.

4.2 KPML Reports

When the user enters key press(es) that match a <regex> tag, the User Device will issue a report.

After reporting, the interpreter terminates the KPML session unless the subscription has a persistence indicator. If the subscription does not have a persistence indicator, the User Device MUST set the state of the subscription to "terminated" in the NOTIFY report.

If the subscription does not have a persistence indicator, to collect more digits the requestor must issue a new request.

NOTE: This highlights the "one shot" nature of KPML, reflecting the balance of features and ease of implementing an interpreter. If your goal is to build an IVR session, we strongly suggest you investigate more appropriate technologies.

KPML reports have two mandatory attributes, code and text. These attributes describe the state of the KPML interpreter on the User Device. Note the KPML status code is not necessarily related to the SIP result code. An important example of this is where a legal SIP subscription request gets a normal SIP 200 OK followed by a NOTIFY, but there is something wrong with the KPML request. In this case, the NOTIFY would include the KPML status code in the KPML report. Note that from a SIP perspective, the SUBSCRIBE and NOTIFY were successful. Also, if the KPML failure is not recoverable, the User Device will most likely set the Subscription-Sate to "terminated". This lets the SIP machinery know the subscription is no longer active.

4.2.1 Pattern Match Reports

If a pattern matches, the User Device will emit a KPML report. Since this is a success report, the code is "200" and the text is "OK".

The KPML report includes the actual digits matched in the digit attribute. The digit string uses the conventional characters '*' and '#' for star and octothorpe respectively. The KPML report also includes the tag attribute if the regex that matched the digits had a tag attribute.

If the subscription requested digit suppression (Section 4.1.3) and the User Device suppressed digits, the suppressed attribute indicates "true". The default value of suppressed is "false".

NOTE: KPML does not include a timestamp. There are a number of reasons for this. First, what timestamp would in include? Would

it be the time of the first detected key press? The time the interpreter collected the entire string? A range? Second, if the RTP timestamp is a datum of interest, why not simply get RTP in the first place? That all said, if it is really compelling to have the timestamp in the response, it could be an attribute to the <response> tag.

4.2.2 KPML No Match Reports

There are a few circumstances in which the User Device will emit a no match report. They are an immediate NOTIFY in response to SUBSCRIBE request (no digits detected yet), a request for service not supported by User Device, or a failure of a digit map to match a string (timeout).

4.2.2.1 Immediate NOTIFY

The NOTIFY in response to a SUBSCRIBE request has no KPML if there are no matching buffered digits. An example of this is in Figure 14.

If there are buffered digits in the SUBSCRIBE request that match a pattern, then the NOTIFY message in response to the SUBSCRIBE request MUST include the appropriate KPML document.

```
NOTIFY sip:application@example.com SIP/2.0
Via: SIP/2.0/UDP proxy.example.com
Max-Forwards: 70
To: <sip:application@example.com>
From: <sip:endpoint@example.net>
Call-Id: 439hu409h4h09903fj0ioij
Subscription-State: active; expires=7200
CSeq: 49851 NOTIFY
Event: kpml
```

Figure 14: Immediate NOTIFY Example

5. DRegex

5.1 Overview

This subsection is informative in nature.

The Digit REGular EXpression (DRegex) syntax is a telephony-oriented mapping of POSIX Extended Regular Expressions (ERE) [17].

KPML does not use full POSIX ERE for the following reasons.

- KPML will often run on high density or extremely low power and memory footprint devices.
- Telephony application convention uses the star symbol ("*") for the star key and "x" for any digit 0-9. Requiring the developer to escape the star ("*") and expand the "x" ("[0-9]") is error prone. This also leads DRegex to using the dot (".") to indicate repetition, which was the function of the unadorned star in POSIX ERE.
- POSIX ERE has clear, unambiguous rules for the precedence of the alternation operator ("|"). However, a few people in the SIPPING Work Group thought we should not allow them. This was due to implementers not getting precedence right in MGCP [15] and H.248.1 [16].

The following table shows the mapping from DRegex to POSIX ERE.

DRegex	POSIX ERE
*	*
.	*
x	[0-9]
[xc]	[0-9c]

Table 1: DRegex to POSIX ERE Mapping

The first substitution, which replaces a star for an escaped star, is because telephony application designers are used to using the star for the (very common) star key. Requiring an escape sequence for this common pattern would be error prone. In addition, the usage found in DRegex is the same as found in MGCP [15] and H.248.1 [16].

Likewise, the use of the dot instead of star is common usage from MGCP and H.248.1, and reusing the star in this context would also be confusing and error prone.

The "x" character is a common indicator of a dialed digit. We use it here, continuing the convention.

Users need to take care not to confuse the DRegex syntax with POSIX EREs. They are NOT identical. In particular there are many features of POSIX EREs that DRegex does not support.

As an implementation note, if one makes the substitutions described in the above table, then a standard POSIX ERE engine can parse the digit string. However, the mapping does not work in the reverse

(POSIX ERE to DRegex) direction. DRegex only implements the Normative behavior described below.

5.2 Operation

White space is removed before parsing DRegex. This enables sensible pretty printing in XML without affecting the meaning of the DRegex string.

The following rules demonstrate the use of DRegex in KPML.

Entity	Matches
character	digits 0-9 and A-D (case insensitive)
*	*
#	#
[character selector]	Any character in selector
[^digit selector]	Any digit (0-9) NOT in selector
[range1-range2]	Any digit (0-9) in range from range1 to range2, inclusive
x	Any digit 0-9
{m}	m repetitions of previous pattern
{m,}	m or more repetitions of previous pattern
{,n}	At most n (including zero) repetitions of previous pattern
{m,n}	at least m and at most n repetitions of previous pattern
Lc	Match the character c if it is "long"; c is a digit 0-9 and A-D, #, or *.

Example	Description
1	Matches the digit 1
[179]	Matches 1, 7, or 9
[^01]	Matches 2, 3, 4, 5, 6, 7, 8, 9
[2-9]	Matches 2, 3, 4, 5, 6, 7, 8, 9
x	Matches 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
*6[179#]	Matches *61, *67, *69, or *6#
x{10}	Ten digits (0-9)
011x{7,15}	011 followed by seven to fifteen digits
L*	Long star

6. Formal Syntax

6.1 DRegex

The following definition follows RFC2234 [2]. The definition of DIGIT is from the CORE specification of RFC2234, namely the characters "0" through "9". Note the DRegexCharacater is not a HEXDIG from RFC2234. In particular, DRegexCharacter neither includes "E" nor "F". Moreover DRegexCharacter is case insensitive, unlike HEXDIG.

```
DRegex           = 1*( DRegexPosition [ RepeatCount ] )
DRegexPosition   = DRegexSymbol / DRegexSet
DRegexSet        = ( "[" DRegexSetList "]" ) /
                   ( "[^" DigitList "]" )
DRegexSetList    = 1*( (DIGIT "-" DIGIT) / DRegexSymbol )
DigitList        = 1*( (DIGIT "-" DIGIT) / DIGIT )
DRegexSymbol     = DRegexCharacter / ( "L" DRegexCharacter )
RepeatCount      = "." / "{" RepeatRange "}"
RepeatRange      = Count / ( Count "," Count ) /
                           ( Count "," ) / ( "," Count )
Count            = 1*(DIGIT)
DRegexCharacter  = DIGIT / "*" / "#" / "A" / "a" / "B" / "b" /
                           "x" / "X" / "C" / "c" / "D" / "d"
```

Note that future extensions to this document may introduce other characters for DRegexCharacter, in the scheme of H.248.1 [16] or possibly as named strings or XML namespaces.

6.2 KPML Request

The following syntax for KPML requests uses the XML Schema [8].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com)
     by Eric Burger (Brooktrout Technology, Inc.) -->
<xs:schema targetNamespace="urn:ietf:params:xml:ns:kpml-request"
 xmlns="urn:ietf:params:xml:ns:kpml-request"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">
  <xs:element name="kpml-request">
    <xs:annotation>
      <xs:documentation>IETF Keypad Markup Language Request
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="stream" minOccurs="0">
          <xs:complexType>
            <xs:choice>
              <xs:element name="reverse" minOccurs="0"/>
              <xs:any namespace="##other"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <xs:element name="pattern">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="flush" minOccurs="0">
                <xs:annotation>
                  <xs:documentation>
                    Default is to not flush buffer
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string"/>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="regex" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    Key press notation is a string to allow
                    for future extension of non-16 digit
                    keypads or named keys
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType mixed="true">
                  <xs:choice>
```



```
                    <xs:element name="pre" minOccurs="0">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string"/>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##other"/>
                  </xs:choice>
                  <xs:attribute name="tag" type="xs:string"
                                use="optional"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="persist" use="optional">
              <xs:annotation>
                <xs:documentation>Default is "one-shot"
                </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="one-shot"/>
                  <xs:enumeration value="persist"/>
                  <xs:enumeration value="single-notify"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="interdigittimer"
                          type="xs:integer"
                          use="optional">
              <xs:annotation>
                <xs:documentation>Default is 4000 (ms)
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="criticaldigittimer"
                          type="xs:integer"
                          use="optional">
              <xs:annotation>
                <xs:documentation>Default is 1000 (ms)
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="extradigittimer"
                          type="xs:integer"
                          use="optional">
              <xs:annotation>
                <xs:documentation>Default is 500 (ms)
```

```
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="long" type="xs:integer"
                          use="optional"/>
            <xs:attribute name="longrepeat" type="xs:boolean"
                          use="optional"/>
            <xs:attribute name="nopartial" type="xs:boolean"
                          use="optional">
              <xs:annotation>
                <xs:documentation>Default is false
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="enterkey" type="xs:string"
                          use="optional">
              <xs:annotation>
                <xs:documentation>No default enterkey
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:string"
                    use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 16: XML Schema for KPML Requests

6.3 KPML Response

The following syntax for KPML responses uses the XML Schema [8].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com)
     by Eric Burger (Brooktrout Technology, Inc.) -->
<xs:schema targetNamespace="urn:ietf:params:xml:ns:kpml-response"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="urn:ietf:params:xml:ns:kpml-response"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">
  <xs:element name="kpml-response">
    <xs:annotation>
      <xs:documentation>IETF Keypad Markup Language Response
```



```
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="version" type="xs:string"
                    use="required"/>
      <xs:attribute name="code" type="xs:string"
                    use="required"/>
      <xs:attribute name="text" type="xs:string"
                    use="required"/>
      <xs:attribute name="suppressed" type="xs:boolean"
                    use="optional"/>
      <xs:attribute name="forced_flush" type="xs:string"
                    use="optional">
        <xs:annotation>
          <xs:documentation>
            String for future use for e.g., number of digits lost.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="digits" type="xs:string"
                    use="optional"/>
      <xs:attribute name="tag" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Matches tag from regex in request
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

7. Enumeration of KPML Status Codes

KPML status codes broadly follow their SIP counterparts. Codes that start with a 2 indicate success. Codes that start with a 4 indicate failure. Codes that start with a 5 indicate a server failure, usually a failure to interpret the document or to support a requested feature.

KPML clients MUST be able to handle arbitrary status codes by examining the first digit only.

Any text can be in a KPML report document. KPML clients MUST NOT interpret the text field.

Code	Text
200	Success
402	User Terminated Without Match
423	Timer Expired
481	Dialog Not Found
487	Subscription Expired
501	Bad Document
502	Namespace Not Supported
531	Persistent Subscriptions Not Supported
532	Multiple Regular Expressions Not Supported
533	Multiple Subscriptions on a Dialog Not Supported

Table 4: KPML Status Codes

8. IANA Considerations

This document registers a new SIP Event Package, two new MIME types, and two new XML namespaces.

8.1 SIP Event Package Registration

Package name: kpml
Type: package
Contact: Eric Burger, <e.burger@ieee.org>
Published Specification: RFCXXXX

8.2 MIME Media Type application/kpml-request+xml

MIME media type name:	application
MIME subtype name:	kpml-request+xml
Required parameters:	none
Optional parameters:	Same as charset parameter application/xml as specified in XML Media Types [3]
Encoding considerations:	See RFC3023 [3].
Security considerations:	See Section 10 of RFC3023 [3] and Section 9 of RFCXXXX
Interoperability considerations:	See RFC2023 [3] and RFCXXXX
Published specification:	RFCXXXX
Applications which use this media type:	Session-oriented applications that have primitive user interfaces.
Personal and email address for further information:	Eric Burger <e.burger@ieee.org>
Intended usage:	COMMON

Additional Information:
Magic Number: None
File Extension: .xml
Macintosh file type code: "TEXT"

8.3 MIME Media Type application/kpml-response+xml

MIME media type name:	application
MIME subtype name:	kpml-resposne+xml
Required parameters:	none
Optional parameters:	Same as charset parameter application/xml as specified in XML Media Types [3]
Encoding considerations:	See RFC3023 [3].
Security considerations:	See Section 10 of RFC3023 [3] and Section 9 of RFCXXXX
Interoperability considerations:	See RFC2023 [3] and RFCXXXX
Published specification:	RFCXXXX
Applications which use this media type:	Session-oriented applications that have primitive user interfaces.
Personal and email address for further information:	Eric Burger <e.burger@ieee.org>
Intended usage:	COMMON

Additional Information:
Magic Number: None
File Extension: .xml
Macintosh file type code: "TEXT"

8.4 URN Sub-Namespace Registration for urn:ietf:xml:ns:kpml-request

URI: urn:ietf:params:xml:ns:kpml-request

Registrant Contact: IETF, SIPPING Work Group <sipping@ietf.org>, Eric Burger <e.burger@ieee.org>.

XML:

```
<?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>Key Press Markup Language Request</title>
  </head>
  <body>
    <h1>Namespace for Key Press Markup Language Request</h1>
    <h2>urn:ietf:params:xml:ns:kpml-request</h2>
    <p>
<a href="ftp://ftp.rfc-editor.org/in-notes/rfcXXXX.txt">RFCXXXX</a>.
    </p>
  </body>
</html>
```

8.5 URN Sub-Namespace Registration for urn:ietf:xml:ns:kpml-response

URI: urn:ietf:params:xml:ns:kpml-response

Registrant Contact: IETF, SIPPING Work Group <sipping@ietf.org>, Eric Burger <e.burger@ieee.org>.

XML:

```
<?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>Key Press Markup Language Response</title>
  </head>
  <body>
    <h1>Namespace for Key Press Markup Language Response</h1>
    <h2>urn:ietf:params:xml:ns:kpml-response</h2>
    <p>
<a href="ftp://ftp.rfc-editor.org/in-notes/rfcXXXX.txt">RFCXXXX</a>.
    </p>
  </body>
</html>
```


8.6 KPML Request Schema Registration

Per RFC3688 [7], please register the XML Schema for KPML as referenced in Section 6.2 of RFCXXXX.

URI: Please assign.

Registrant Contact: IETF, SIPPING Work Group <sipping@ietf.org>, Eric Burger <e.burger@ieee.org>.

8.7 KPML Response Schema Registration

Per RFC3688 [7], please register the XML Schema for KPML as referenced in Section 6.3 of RFCXXXX.

URI: Please assign.

Registrant Contact: IETF, SIPPING Work Group <sipping@ietf.org>, Eric Burger <e.burger@ieee.org>.

9. Security Considerations

As an XML markup, all of the security considerations of RFC3023 [3] and RFC3406 [6] must be met. Pay particular attention to the robustness requirements of parsing XML.

Key press information is potentially sensitive. For example, it can represent credit card, calling card, or other personal information. Hijacking sessions allow unauthorized entities access to this sensitive information. Therefore, signaling SHOULD be secure, e.g., use of TLS and sips: SHOULD be used. Moreover, the information itself is sensitive, therefore the use of S/MIME or other appropriate mechanism SHOULD be used.

Subscriptions MUST be authenticated.

User Devices MUST support digest authentication.

User Devices MUST support the sips: scheme and TLS.

User Devices MUST NOT buffer USER input prior to an authenticated subscription.

User Devices MUST buffer User Input upon receipt of an authenticated and accepted subscription.

User Devices implementing this specification MUST implement TLS and SHOULD implement S/MIME at a minimum.

10. Examples

This section is informative in nature. If there is a discrepancy between this section and the normative sections above, the normative sections take precedence.

10.1 Monitoring for Octothorpe

A common need for pre-paid and personal assistant applications is to monitor a conversation for a signal indicating a change in user focus from the party they called through the application to the application itself. For example, if you call a party using a pre-paid calling card and the party you call redirects you to voice mail, digits you press are for the voice mail system. However, many applications have a special key sequence, such as the octothorpe (#, or pound sign) or *9 that terminate the called party session and shift the user's focus to the application.

Figure 20 shows the KPML for long octothorpe.

```
<?xml version="1.0">
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern>
    <regex>L#</regex>
  </pattern>
</kpml-request>
```

Figure 20: Long Octothorpe Example

The regex value L indicates the following digit needs to be a long-duration key press.

10.2 Dial String Collection

In this example, the User Device collects a dial string. The application uses KPML to quickly determine when the user enters a target number. In addition, KPML indicates what type of number the user entered.


```
<?xml version="1.0">
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern>
    <regex tag="local-operator">0</regex>
    <regex tag="ld-operator"/>00</regex>
    <regex tag="vpn">7[x][x][x]</regex>
    <regex tag="local-number7">9xxxxxxx</regex>
    <regex tag="RI-number">9401xxxxxxx</regex>
    <regex tag="local-number10">9xxxxxxxxxx</regex>
    <regex tag="ddd">91xxxxxxxxxx</regex>
    <regex tag="iddd">011x.</regex>
  </pattern>
</kpml-request>
```

Figure 21: Dial String KPML Example Code

Note the use of the "tag" attribute to indicate which regex matched the dialed string. The interesting case here is if the user entered "94015551212". This string matches both the "9401xxxxxxx" and "9xxxxxxxxxx" regular expressions. By following the rules described in Section 4.1.5, the KPML interpreter will pick the "9401xxxxxxx" string, as it occurs first in document order (both expressions match the same length). Figure 22 shows the response.

```
<?xml version="1.0"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-resposne"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
      version="1.0"
      code="200" text="OK"
      digits="94015551212" tag="RI-number"/>
```

Figure 22: Dial String KPML Response

11. Call Flow Examples

11.1 Supplemental Digits

This section gives a non-normative example of an application that collects supplemental digits. Supplemental digit collection is where the network requests additional digits after the caller enters the destination address. A typical supplemental dial string is four

digits in length.

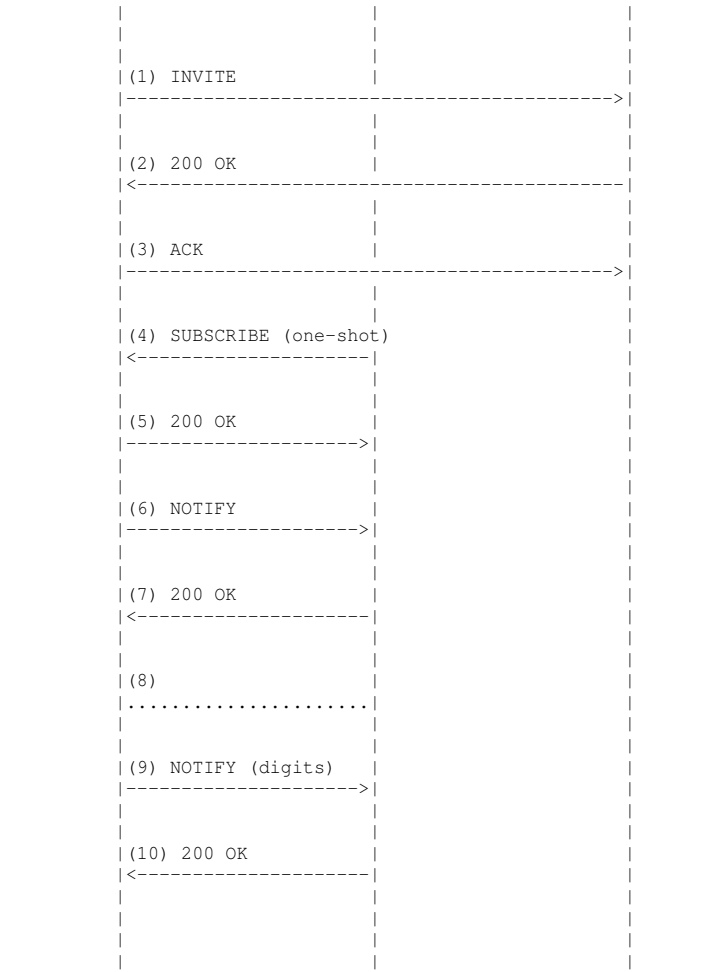


Figure 23: Supplemental Digits Call Flow

In messages (1-3), the ingress gateway establishes a dialog with an egress gateway. The application learns the dialog ID through out-of-band mechanisms, such as the Dialog Package or being coresident with the egress gateway. Part of the ACK message is below, to illustrate the dialog identifiers.

```
ACK sip:gw@subA.example.com SIP/2.0
Via: ...
Max-Forwards: ...
Route: ...
From: <sip:phn@example.com>;tag=jfh21
To: <sip:gw@subA.example.com>;tag=onjwe2
Call-ID: 12345592@subA.example.com
...
```

In message (4), the application requests the gateway collect a string of four key presses.

```
SUBSCRIBE sip:gw@subA.example.com SIP/2.0
Via: SIP/2.0/TCP client.subB.example.com;branch=q4i9ufr4ui3
From: <sip:ap@subB.example.com>;tag=567890
To: <sip:gw@subA.example.com>
Call-ID: 12345601@subA.example.com
CSeq: 1 SUBSCRIBE
Contact: <sip:ap@client.subB.example.com>
Max-Forwards: 70
Event: kpml ;remote-tag="<sip:phn@example.com;tag=jfh21>"
            ;local-tag="sip:gw@subA.example.com;tag=onjwe2"
            ;call-id="12345592@subA.example.com"
Expires: 7200
Accept: application/kpml-response+xml
Content-Type: application/kpml-request+xml
Content-Length: 292

<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
      "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
    version="1.0">
  <pattern persist="one-shot">
    <regex>xxxx</regex>
  </pattern>
</kpml-request>
```

Message (5) is the acknowledgement of the subscription request.

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP subB.example.com;branch=q4i9ufr4ui3;
     received=192.168.125.12
From: <sip:ap@subB.example.com>;tag=567890
To: <sip:gw@subA.example.com>;tag=1234567
Call-ID: 12345601@subA.example.com
CSeq: 1 SUBSCRIBE
Contact: <sip:gw27@subA.example.com>
Expires: 3600
Event: kpml
```

Message (6) is the immediate notification of the subscription.

```
NOTIFY sip:ap@client.subB.example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=gw27id4993
To: <sip:ap@subB.example.com>;tag=567890
From: <sip:gw@subA.example.com>;tag=1234567
Call-ID: 12345601@subA.example.com
CSeq: 1000 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: active;expires=3599
Max-Forwards: 70
Content-Length: 0
```

Message (7) is the acknowledgment of the notification message.

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP subA.example.com;branch=gw27id4993
To: <sip:ap@subB.example.com>;tag=567890
From: <sip:gw@subA.example.com>;tag=1234567
Call-ID: 12345601@subA.example.com
CSeq: 1000 NOTIFY
```

Some time elapses (8).

The user enters the input. The device provides the notification of the collected digits in message (9). Since this was a one-shot subscription, note the Subscription-State is "terminated".


```
NOTIFY sip:ap@client.subB.example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=gw27id4993
To: <sip:ap@subB.example.com>;tag=567890
From: <sip:gw@subA.example.com>;tag=1234567
Call-ID: 12345601@subA.example.com
CSeq: 1001 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: terminated
Max-Forwards: 70
Content-Type: application/kpml-response+xml
Content-Length: 258

<?xml version="1.0" encoding="UTF-8"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-response"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
      version="1.0"
      code="200" text="OK"
      digits="4336"/>
```

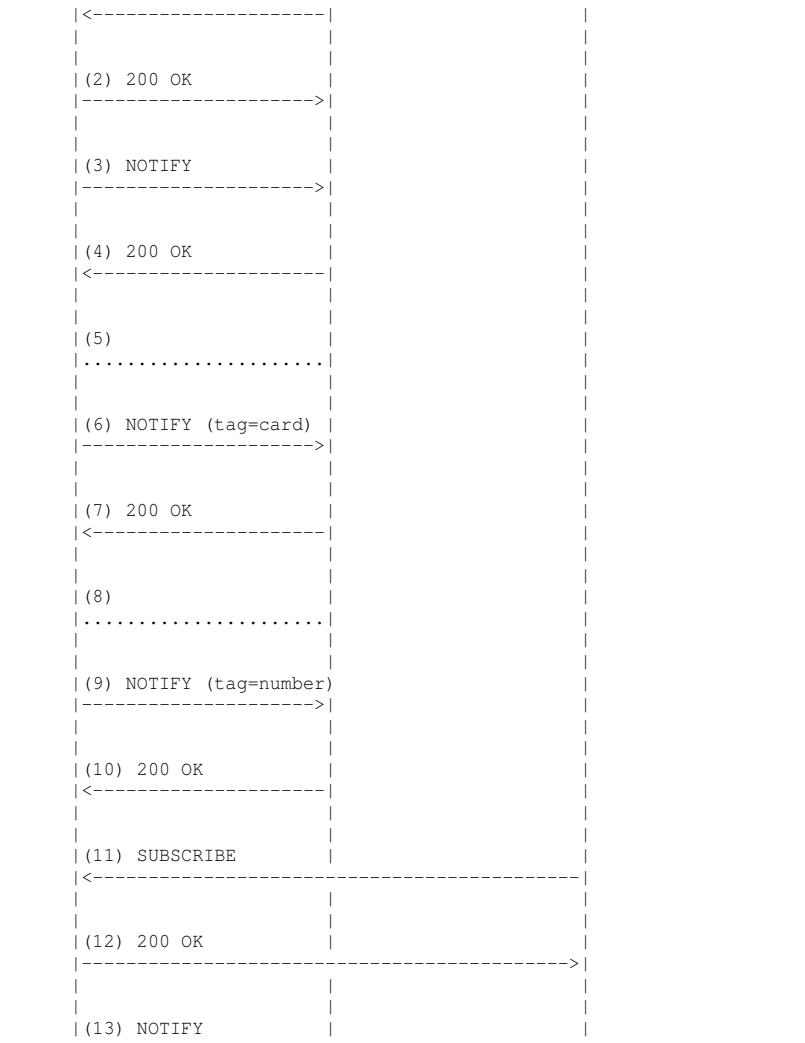
Message (10) is the acknowledgement of the notification.

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP subA.example.com;branch=gw27id4993
To: <sip:ap@subB.example.com>;tag=567890
From: <sip:gw@subA.example.com>;tag=1234567
Call-ID: 12345601@subA.example.com
CSeq: 1001 NOTIFY
```

11.2 Multiple Applications

This section gives a non-normative example of multiple applications. One application collects a destination number to call. That application then waits for a "long pound." During the call, the call goes to a personal assistant application, which interacts with the user. In addition, the personal assistant application looks for a "short pound."

For clarity, we do not show the INVITE dialogs.


(2) 200 OK
(3) NOTIFY
(4) 200 OK
(5)
(6) NOTIFY (tag=card)
(7) 200 OK
(8)
(9) NOTIFY (tag=number)
(10) 200 OK
(11) SUBSCRIBE
(12) 200 OK
(13) NOTIFY

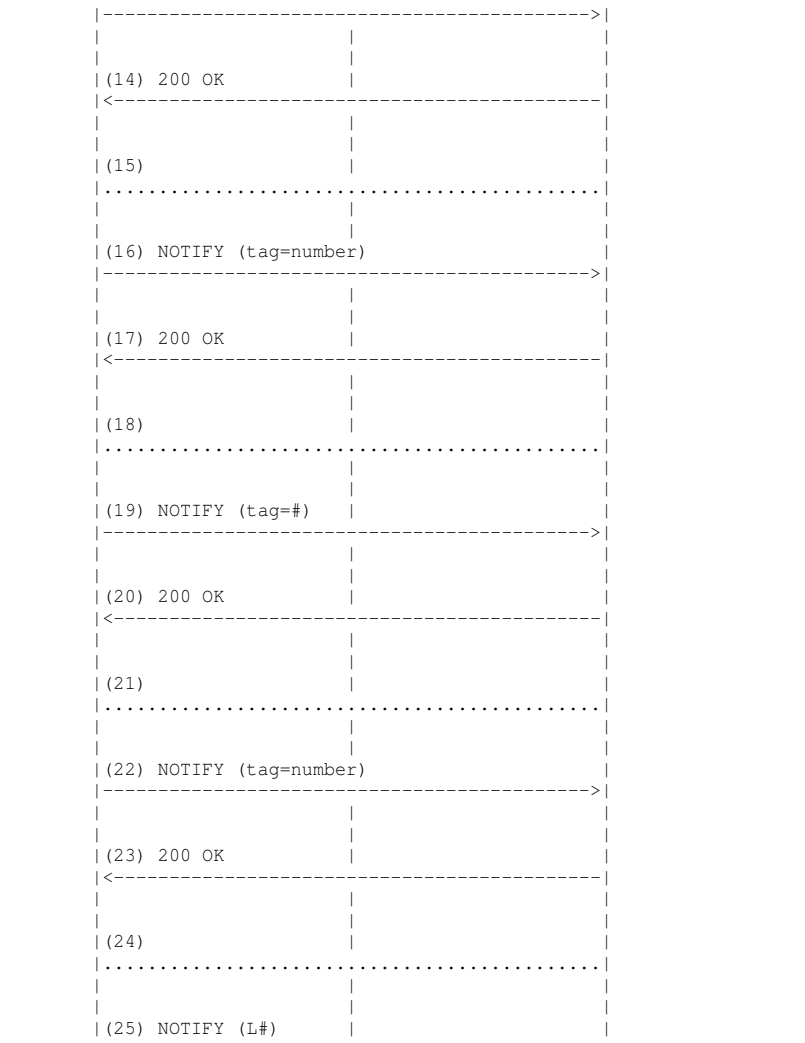

(14) 200 OK
(15)
(16) NOTIFY (tag=number)
(17) 200 OK
(18)
(19) NOTIFY (tag=#)
(20) 200 OK
(21)
(22) NOTIFY (tag=number)
(23) 200 OK
(24)
(25) NOTIFY (L#)

Figure 31: Multiple Application Call Flow

Message (1) is the subscription request for the card number.

```
SUBSCRIBE sip:gw@subA.example.com SIP/2.0
Via: SIP/2.0/TCP client.subB.example.com;branch=3qo3j0ouq
From: <sip:ap@subB.example.com>;tag=978675
To: <sip:gw@subA.example.com>
Call-ID: 12345601@subA.example.com
CSeq: 20 SUBSCRIBE
Contact: <sip:ap@client.subB.example.com>
Max-Forwards: 70
Event: kpml ;remote-tag="<sip:phn@example.com;tag=jfi23>"
            ;local-tag="sip:gw@subA.example.com;tag=oi43jfq"
            ;call-id="12345598@subA.example.com"
Expires: 7200
Accept: application/kpml-response+xml
Content-Type: application/kpml-request+xml
Content-Length: 339

<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern persist="persist">
    <regex tag="card">x{16}</regex>
    <regex tag="number">x{10}</regex>
  </pattern>
</kpml-request>
```

Messages 2-4 are not shown for brevity. Message (6) is the notification of the card number.


```
NOTIFY sip:ap@client.subB.example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=3qo3j0ouq
To: <sip:ap@subB.example.com>;tag=978675
From: <sip:gw@subA.example.com>;tag=9783453
Call-ID: 12345601@subA.example.com
CSeq: 3001 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: active;expires=3442
Max-Forwards: 70
Content-Type: application/kpml-response+xml
Content-Length: 271

<?xml version="1.0" encoding="UTF-8"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-response"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
      version="1.0"
      code="200" text="OK"
      digits="9999888877776666"/>
```

Message (7) is the acknowledgement of the notification. Time goes by in (8). Message (9) is the notification of the dialed number.

```
NOTIFY sip:ap@client.subB.example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=3qo3j0ouq
To: <sip:ap@subB.example.com>;tag=978675
From: <sip:gw@subA.example.com>;tag=9783453
Call-ID: 12345601@subA.example.com
CSeq: 3001 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: active;expires=3542
Max-Forwards: 70
Content-Type: application/kpml-response+xml
Content-Length: 278

<?xml version="1.0" encoding="UTF-8"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-response"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
      version="1.0"
      code="200" text="OK"
      digits="2225551212" tag="number"/>
```

Message (11) is the request for long-pound monitoring.

```
SUBSCRIBE sip:gw@subA.example.com SIP/2.0
Via: SIP/2.0/TCP client.subB.example.com;branch=3qo3j0ouq
From: <sip:ap@subB.example.com>;tag=978675
To: <sip:gw@subA.example.com>
Call-ID: 12345601@subA.example.com
CSeq: 21 SUBSCRIBE
Contact: <sip:ap@client.subB.example.com>
Max-Forwards: 70
Event: kpml ;remote-tag="<sip:phn@example.com;tag=jfi23>"
            ;local-tag="sip:gw@subA.example.com;tag=oi43jfq"
            ;call-id="12345598@subA.example.com"
Expires: 7200
Accept: application/kpml-response+xml
Content-Type: application/kpml-request+xml
Content-Length: 295

<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern persist="single-notify">
    <regex>L#</regex>
  </pattern>
</kpml-request>
```

Message (13) is the request from the personal assistant application for number and pound sign monitoring.


```
SUBSCRIBE sip:gw@subA.example.com SIP/2.0
Via: SIP/2.0/TCP pahost.example.com;branch=xzvsadf
From: <sip:pa@example.com>;tag=4rgj0f
To: <sip:gw@subA.example.com>
Call-ID: 93845@pahost.example.com
CSeq: 21 SUBSCRIBE
Contact: <sip:pa12@pahost.example.com>
Max-Forwards: 70
Event: kpml ;remote-tag="<sip:phn@example.com;tag=jfi23>"
            ;local-tag="sip:gw@subA.example.com;tag=oi43jfq"
            ;call-id="12345598@subA.example.com"
Expires: 7200
Accept: application/kpml-response+xml
Content-Type: application/kpml-request+xml
Content-Length: 332

<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
      version="1.0">
  <pattern persist="persist">
    <regex tag="number">x{10}</regex>
    <regex tag="#">#</regex>
  </pattern>
</kpml-request>
```

Message (18) is the notification of the number collected.

```
NOTIFY sip:pa@example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=xzvsadf
To: <sip:pa@example.com>;tag=4rgj0f
From: <sip:gw@subA.example.com>;tag=9788823
Call-ID: 93845@pahost.example.com
CSeq: 3021 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: active;expires=3540
Max-Forwards: 70
Content-Type: application/kpml-response+xml
Content-Length: 278

<?xml version="1.0" encoding="UTF-8"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-response"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
```

```
      version="1.0"
      code="200" text="OK" digits="3335551212" tag="number"/>
```

Message (21) is the notification of pound sign detected.

```
NOTIFY sip:pa@example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=xzvsadf
To: <sip:pa@example.com>;tag=4rgj0f
From: <sip:gw@subA.example.com>;tag=9788823
Call-ID: 93845@pahost.example.com
CSeq: 3022 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: active;expires=3540
Max-Forwards: 70
Content-Type: application/kpml-response+xml
Content-Length: 264

<?xml version="1.0" encoding="UTF-8"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-response"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
      version="1.0"
      code="200" text="OK"
      digits="#" tag="#"/>
```

Message (27) is the notification of long pound to the card application.


```
NOTIFY sip:ap@client.subB.example.com SIP/2.0
Via: SIP/2.0/UDP subA.example.com;branch=3qo3j0ouq
To: <sip:ap@subB.example.com>;tag=978675
From: <sip:gw@subA.example.com>;tag=9783453
Call-ID: 12345601@subA.example.com
CSeq: 3037 NOTIFY
Contact: <sip:gw27@subA.example.com>
Event: kpml
Subscription-State: active;expires=3216
Max-Forwards: 70
Content-Type: application/kpml-response+xml
Content-Length: 256

<?xml version="1.0" encoding="UTF-8"?>
<kpml-response xmlns="urn:ietf:params:xml:ns:kpml-response"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "urn:ietf:params:xml:ns:kpml-response kpml-response.xsd"
      version="1.0"
      code="200" text="OK"
      digits="#"/>
```

12. References

12.1 Normative References

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

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

[3] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[4] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

[6] Daigle, L., van Gulik, D., Iannella, R. and P. Faltstrom, "Uniform Resource Names (URN) Namespace Definition Mechanisms", BCP 66, RFC 3406, October 2002.

[7] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January

2004.

[8] Thompson, H., Beech, D., Maloney, M. and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC REC-xmlschema-1-20010502, May 2001.

12.2 Informative References

[9] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[10] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

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

[12] Schulzrinne, H. and S. Petrack, "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals", RFC 2833, May 2000.

[13] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[14] Olson, S., Camarillo, G. and A. Roach, "Support for IPv6 in Session Description Protocol (SDP)", RFC 3266, June 2002.

[15] Andreasen, F. and B. Foster, "Media Gateway Control Protocol (MGCP) Version 1.0", RFC 3435, January 2003.

[16] Groves, C., Pantaleo, M., Anderson, T. and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[17] Institute of Electrical and Electronics Engineers, "Information Technology - Portable Operating System Interface (POSIX) - Part 1: Base Definitions, Chapter 9", IEEE Standard 1003.1, June 2001.

[18] "Packet-based Multimedia Communications Systems (includes Annex C - H.323 on ATM)", ITU-T Recommendation H.323v3, September 1999.

[19] International Telecommunications Union, "CONTROL PROTOCOL FOR MULTIMEDIA COMMUNICATION", ITU Recommendation H.245, 1998.

[20] World Wide Web Consortium, "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C Working Draft , April 2002,

http://www.w3.org/TR/voicexml20/>.

[21] Bray, T., Paoli, J., Sperberg-McQueen, C. and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C REC REC-xml-20001006, October 2000.

[22] Hunt, A. and S. McGlashan, "Speech Recognition Grammar Specification Version 1.0", W3C CR CR-speech-grammar-20020626, June 2002.

[23] Rosenberg, J., "A Framework for Application Interaction in the Session Initiation Protocol (SIP)", draft-ietf-sipping-app-interaction-framework-01 (work in progress), February 2004.

[24] Burger (Ed.), E., Van Dyke, J. and A. Spitzer, "Basic Network Media Services with SIP", draft-burger-sipping-netann-08 (work in progress), February 2004.

[25] Burger, E., Van Dyke, J. and A. Spitzer, "Media Server Control Markup Language (MSCML) and Protocol", draft-vandyke-mscml-04 (work in progress), March 2004.

[26] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-01 (work in progress), February 2004.

[27] Rosenberg, J. and H. Schulzrinne, "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP", draft-ietf-sipping-dialog-package-02 (work in progress), June 2003.

Authors' Addresses

Eric Burger
Brooktrout Technology, Inc.
18 Keewaydin Dr.
Salem, NH 03079
USA

EMail: e.burger@ieee.org

Martin Dolly
AT&T Labs

EMail: mdolly@att.com

Appendix A. Contributors

Ophir Frieder of the Illinois Institute of Technology collaborated on the development of the buffer algorithm.

Jeff Van Dyke worked enough hours and wrote enough text to be considered an author under the old rules.

Robert Fairlie-Cuninghame, Cullen Jennings, Jonathan Rosenberg, and I were the members of the Application Stimulus Signaling Design Team. All members of the team contributed to this work. In addition, Jonathan Rosenberg postulated DML in his "A Framework for Stimulus Signaling in SIP Using Markup" draft.

We liberally adopted text from Rohan Mahy's Signaled Telephony Events draft for the motivation section.

This version of KPML has significant influence from MSCML, the SnowShore Media Server Control Markup Language. Jeff Van Dyke and Andy Spitzer were the primary contributors to that effort.

That said, any errors, misinterpretation, or fouls in this document are our own.

Appendix B. Acknowledgements

Hal Purdy and Eric Cheung of AT&T Laboratories helped immensely through many conversations and challenges.

Steve Fisher of AT&T Laboratories suggested the digit suppression syntax and provided excellent review of the document.

Terence Lobo of SnowShore Networks made it all work.

Jerry Kamitses, Swati Dhuleshia, Shaun Bharrat, Sunil Menon, and Bryan Hill helped with clarifying the buffer behavior and DRegex syntax.

Silvano Brewster and Bill Fenner of AT&T Laboratories helped considerably with makeing the text clear and DRegex tight.

Bert Culpepper and Allison Manking gave an early version of this document a good scouring.

Rohan Mahy gave Martin and I considerable moral support in the production of this document.

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 procedures with respect to rights in RFC 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.

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

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.

Internet Engineering Task Force James M. Polk
Internet Draft Cisco Systems
Expiration: Jan 19th, 2005 Brian Rosen
File: draft-ietf-sipping-location-requirements-01.txt Marconi

Requirements for Session Initiation Protocol Location Conveyance

July 19, 2004

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/lid-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document presents the framework and requirements for usage of the Session Initiation Protocol (SIP) [RFC 3261] to convey user location information from a Session Initiation Protocol (SIP) user agent to another SIP entity. We consider cases where location information is conveyed from end to end, as well as cases where message routing by intermediaries is influenced by the location of the session initiator. We offer a set of solutions to the requirements, based on the scenario(s) being addressed.

Table of Contents

1. Introduction

This document presents the framework and requirements for the usage of the Session Initiation Protocol (SIP) [1] for conveyance of user location information object described by [7] from a SIP User Agent to another SIP entity.

There are several situations in which it is appropriate for SIP to be used to convey Location Information (LI) from one SIP entity to another. This document specifies requirements when a SIP UAC knows its location by some means not specified herein, and needs to inform another SIP entity. One example is to reach your nearest pizza parlor. A chain of pizza parlors may have a single well known uri (sip:pizzaparlor.com), that is forwarded to the closest franchise by the pizzaparlor.com proxy server. The receiving franchise UAS uses the location information of the UAC to schedule your delivery.

Another important example is emergency calling. A call to

sip:sos@example.com is an emergency call as in [3]. The example.com proxy server must route the call to the correct emergency response center (ERC) determined by the location of the caller. At the ERC, the UAS must determine the correct police/fire/ambulance/... service, which is also based on your location. In many jurisdictions, accurate location information of the caller in distress is a required component of a call to an emergency center. A third example is a direction service, which might give you verbal directions to a venue from your present position. This is a case where only the destination UAS needs to receive the location information.

This document does not discuss how the UAC discovers or is configured with its location (either coordinate or civic based). It also does not discuss the contents of the Location Object (LO). It does specify the requirements for the "using protocol" in [7]. Sections 7, 8 and 9 give specific examples (in well-formed SIP messages) of SIP UA and Proxy behavior for location conveyance, the last of which is a section devoted to the unique circumstances regarding emergency calling. Section 10 addresses how this document adheres to the requirements specified in [7] (Geopriv Requirements). Sections 11 and 12 list the current open issues with location conveyance in SIP, and the new open issues recently discovered as a result of the added effort to this revision.

1.1 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 [2].

1.2 Changes from Prior Versions

[NOTE TO RFC-EDITOR: If this document is to be published as an RFC, this section is to be removed prior to that event.]

This is a list of the changes that have been made from the -00 working group version of this ID to this version:

- Added the offered solution in detail (with message flows, appropriate SIP Methods for location conveyance, and
- Synchronized the requirements here with those from the Geopriv Working Group's (attempting to eliminate overlap)
- Took on the task of making this effort the SIP "using protocol" specification from Geopriv's POV

- Refined the Open Issues section to reflect the progress we've made here, and to indicate what we have discovered needs addressing, but has not been to date.

This is a list of the changes that have been made from the -01 individual submission version to the WG -00 version of this ID:

- Brian Rosen was brought on as a co-author
- Requirements that a location header were negatively received in the previous version of this document. AD and chair advice was to move all location information into a message body (and stay away from headers)
- Added a section of "emergency call" specific requirements
- Added an Open Issues section to mention what hasn't been resolved yet in this effort

This is a list of the changes that have been made from the individual submission version -00 to the -01 version

- Added the IPR Statement section
- Adjusted a few requirements based on suggestions from the Minneapolis meeting
- Added requirements that the UAC is to include from where it learned its location in any transmission of its LI
- Distinguished the facts (known to date) that certain jurisdictions relieve persons of their right to privacy when they call an ERC, while other jurisdictions maintain a person's right to privacy, while still others maintain a person's right to privacy - but only if they ask that their service be set up that way.
- Made the decision that TLS is the security mechanism for location conveyance in emergency communications (vs. S/MIME, which is still the mechanism for UA-to-UA non-emergency location conveyance cases).
- Added the Open Issue of whether a Proxy can insert location information into an emergency SIP INVITE message, and some of the open questions surrounding the implications of that action
- added a few names to the acknowledgements section

2. In the Body or in a Header

When one user agent wants to inform another user agent where they are, it seems reasonable to have this accomplished by placing the

location information (coordinate or civic) in an S/MIME registered and encoded message body, and sending it as part of a SIP request or response. No routing of the request based on the location information is required in this case; therefore no SIP Proxies between these two UAs need to view the location information contained in the SIP messages.

Although SIP [1} does not permit a proxy server to modify or delete a body, there is no restriction on viewing bodies. However, S/MIME protection implemented on bodies is only specified between UAS and UAC, and if engaged, would render the location object opaque to a proxy server for any desired modification if it is not correct or precise enough from that proxy's point of view (were it to be able to view it). This problem is similar to that raised in Session Policy [8], where an intermediary may need information in a body, such as IP address of media streams or codec choices to route a call properly. Requirements in [8] are applicable to routing based on location, and are incorporated in these requirements by reference.

It is conceivable to create a new header for location information. However, [7] prefers S/MIME for security of Location Information, and indeed S/MIME is preferable in SIP for protecting one part of a message. Accordingly, these requirements specify location be carried in a body.

It is the use of S/MIME however, that limits routing based on location. Therefore, it seems appropriate to require that, where routing is dependent on location, protection of the location information object be accomplished by other mechanisms: here TLS ("sips:" from [1]). It is envisioned that S/MIME SHOULD be used when location information is not required by proxy servers, and TLS MUST be used when it is. The UAC will need to know the difference in the call's intent as to which security mechanism to engage for LI conveyance.

This document does not address the behavior or configuration of SIP Proxy Servers in these cases in order to accomplish location-sensitive routing. That is out of scope, and left for further (complementary) efforts.

3. Scope of Location in a Message Body

As concluded from the previous section, location information is to be contained within a message body. If either another body (SDP for example) is also to be sent in the message, or the LI is to be protected with S/MIME, the rules stated in section 7 of [1] regarding multipart MIME bodies MUST be followed. The format and privacy/security rules of the location information SHOULD be defined within the Geopriv WG.

4. Requirements for UA-to-UA Location Conveyance

The following are the requirements for UA-to-UA Location Conveyance Situations where routing is not based on the LI of either UA:

- U-U1 - MUST work with dialog-initiating SIP Requests and responses, as well as the SIP MESSAGE method [4], and SHOULD work with most SIP messages.
- U-U2 - UAC Location information SHOULD remain confidential in route to the destination UA.
- U-U3 - The privacy and security rules established within the Geopriv Working Group that would categorize SIP as a 'using protocol' MUST be met [7].
- U-U4 - The LI MUST be contained in the LO as defined in [13], which will satisfy all format requirements for interoperability.
- U-U5 - The requirements of a "using protocol" by RFC 3693 (Geopriv Requirements) MUST be met.

5. Requirements for UA-to-Proxy Server Location Conveyance

The following are the requirements for UA-to-Proxy Server Location Conveyance situations:

- U-PS1 - MUST work with dialog-initiating SIP Requests and responses, as well as the SIP MESSAGE method[4], and SHOULD work with most SIP messages.
- U-PS2 - UAC location information SHOULD remain confidential with respect to entities to which the location information is not addressed, but MUST be useable by intermediary proxy servers.
- U-PS3 - The privacy and security rules established within the Geopriv Working Group which would categorize SIP as a 'using protocol' MUST be met [7].
- U-PS4 - Modification or removal of the LO by proxy servers MUST NOT be required (as [1] currently forbids this).
- U-PS5 - any mechanism used to prevent unwanted observation of this Location Information CANNOT fail the SIP Request if not understood by intermediary SIP entities or the destination UAS.

U-PS6 - Proxy Servers that do not or cannot understand the Location Information in the message body for routing purposes MUST NOT fail the SIP Request.

U-PS7 û It MUST be possible for a proxy server to assert the validity of the location information provided by the UA. Alternatively, it is acceptable for there to be a mechanism for a proxy server to assert a location object itself.

6. Additional Requirements for Emergency Calls

Emergency calls have requirements that are not generally important to other uses for location in SIP:

Emergency calls presently have between 2 and 8-second call setup times. There is ample evidence that the longer call setup end of the range causes an unacceptable number of callers to abandon the call before it is completed. Two-second call completion time is a goal of many existing emergency call centers. Allocating 25% of the call set up for processing privacy concerns seems reasonable; 1 second would be 50% of the goal, which seems unacceptable; less than 0.5 second seems unachievable, therefore:

E-1 - Privacy mechanisms MUST add no more than 0.5 second of call setup time when implemented in present technology UAs and Proxy Servers.

It may be acceptable for full privacy mechanisms related to the location of the UAC (and it's user) to be tried on an initial attempt to place a call, as long as the call attempt may be retried without the mechanism if the first attempt fails. Abandoning privacy in cases of failure of the privacy mechanism might be subject to user preference, although such a feature would be within the domain of a UA implementation and thus not subject to standardization. It should be noted that some jurisdictions have laws that explicitly deny any expectation of location privacy when making an emergency call, while others grant the user the ability to remain anonymous even when calling an ERC. So far, this has been offered in some jurisdictions, but the user within that jurisdiction must state this preference, as it is not the default configuration.

E-2 û Privacy mechanisms MUST NOT be mandatory for successful conveyance of location during an (sos-type) emergency call.

E-3 - It MUST be possible to provide a privacy mechanism (that does not violate the other requirements within this document) to a user within a jurisdiction that gives that user the right to choose not to reveal their location even when contacting an ERC.

E-4 û The retention and retransmission policy of the ERC MUST be able to be made available to the user, and override the user's normal policy when local regulation governs such retention and retransmission (but does not violate requirement E-3). As in E-2 above, requiring the use of the ERC's retention and/or retransmission policy may be subject to user preference although in most jurisdictions, local laws specify such policies and may not be overridden by user preference.

Location information is considered so important during emergency calls, that it is to be transmitted even when it is not considered reliable, or might even be wrong. For example, some application might know that the DHCP reply with location information was overwritten recently (or exactly) when a VPN connection was activated. This could, and likely will, provide any new location information to the UA from somewhere far away from the UA (perhaps the user's corporate facility).

E-5 Location information MUST be transmitted, if known to the UAC, in all calls to an ERC, even in the case it is not considered reliable.

With that in mind, it is important to distinguish the location information learned locally from LI learned over a VPN; which in itself is useful additional information to that ERC operator.

E-7 THE UA must provide the actual LI of the endpoint, and not location which might have been erroneously given to it by, e.g. a VPN tunnel DHCP server.

7. Location Conveyance using SIP

Geopriv is the IETF working group assigned to define a Location Object for carrying within another protocol to convey geographic location of an endpoint to another entity. This Location Object will be supplied within SIP to convey location of a UA (or user of a UA). The Location Object (LO) is defined in [13]. Section 26 of [1] defines the security functionality SIPS for transporting SIP messages with either TLS or IPsec, and S/MIME for encrypting message bodies from SIP intermediaries that would otherwise have access to reading the clear-text bodies. For UA-to-UA location conveyance, using the PIDF-LO body satisfies the entire format and message-handling requirements as stated in the baseline Geopriv requirements [7]. SIP entities that will carry an LO MUST IMPLEMENT S/MIME for encrypting on an end-to-end basis the location of a user agent, satisfying [7]'s security requirements. The SIPS-URI from [1] SHOULD also be used for further message protection (message integrity, authentication and message confidentiality) and MUST be used when S/MIME is not used. The entities sending and receiving the LO MUST implement the privacy and security instructions in the

LO.
Self-signed certificates SHOULD NOT be used for protecting LI, as the sender does not have a secure identity of the recipient.
Several LOs MAY be included in a body as long as the message length is less than the maximum permitted for a single message in the network the Location will be conveyed within.
Several SIP Methods are capable (and applicable) to carry the LO. The Methods are divided into two groups, one for those applicable for UA-to-UA location conveyance, and the other group for UA-to-Proxy Location conveyance for routing the message.
The list of applicable Methods for UA-to-UA location conveyance is:

- INVITE,
- UPDATE,
- MESSAGE, and
- PUBLISH.

The list of applicable Methods for UA-to-Proxy location conveyance is:

- INVITE,
- UPDATE, and maybe
- MESSAGE

While the authors do not yet see a reason to have location conveyed in the OPTIONS, ACK, PRACK, BYE, REFER and CANCEL Methods, we do not see a reason to prevent carrying a LO within these Method Requests as long as the SIP message meets the requirements stated within this document.
A 200 OK to an INVITE can carry the UAS's LO back to the UAC that provided their location in the INVITE, but this is not something that can be required due to the timing of the INVITE to 200 OK messages, with potential local/user policy requiring the called user to get involved in determining if the caller is someone they wish to give location to (and at what precision).
There is an open question as to whether the SUBSCRIBE and NOTIFY Methods should be addressed in this document, or another document at a later date. This combination of Methods would be used in SIP by having a UA or SIP Server offering a subscription to another UA for the purposes of location refresh if the subscribed-to UA changes location within a given time interval. This capability is not currently considered critical, and considered "phase II" within the Geopriv working group, but it is an open question here as to whether the SIP/SIPPING WGs want this to be specified here as a behavior (it should be pretty straight forward).

For UA-to-Proxy location conveyance, there are two cases: one in which all proxies on the path from the UA to the proxy that requires location can be trusted with the LI, and one in which intermediate proxies may not be trusted. The former may be implemented with "hop-by-hop" security as specified in [1] using sips: (i.e. TLS security). In particular, emergency call routing requires routing proxies to know location, and sips: protection is appropriate. The latter case is under study by the SIPPING working group under the subject "End to Middle" security [12].
Regardless which scenario (UA-to-UA or UA-to-Proxy) is used to convey location, SIP entities MUST adhere to the rules of [7], specifically the retention and distribution (privacy) attributes of a UA's location. When Alice is deciding how to transmit her location, she should be keenly aware of the parameters in which she wants her location to be stored and distributed. However, once she sends that location information to Bob, he MUST also now obey Alice's wishes regarding these privacy attributes if he is deciding to inform another party about Alice. This is a fundamental principle of the Geopriv Working Group, i.e. "PRIVACY".

8. User Agent-to-User Agent Location Conveyance

The offered solution here for the User-to-User solution for location conveyance between UAs is used with the INVITE, UPDATE, MESSAGE, and PUBLISH Methods in the following subsections.

8.1 UA-to-UA using INVITE Method

Below is a common SIP session set-up sequence between two user agents. In this example, Alice will provide Bob with her geographic location in the INVITE message.

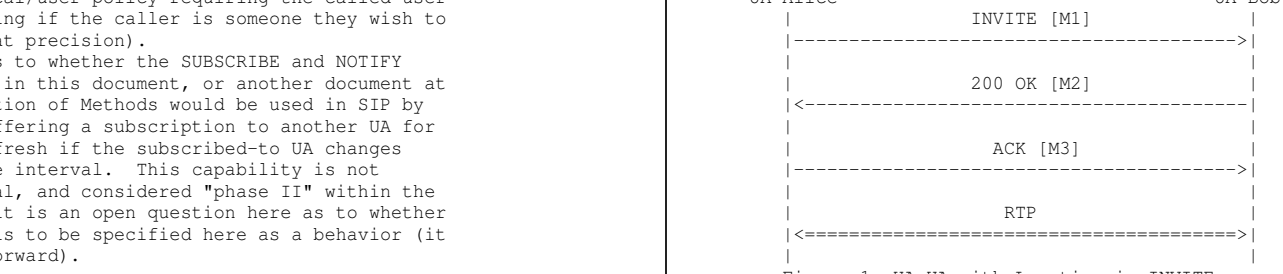


Figure 1. UA-UA with Location in INVITE

User agent Alice INVITEs user agent Bob to a session [M1 of Figure 1]. Within this INVITE is a multipart body indication that it is S/MIME encrypted [according to the rules of 1] by Alice for Bob. One body part contains the SDP offered by Alice to Bob. Alice's location (here coordinate based) is the other body part contained in this INVITE. Bob responses with a 200 OK [M2] (choosing a codec as specified by the Offer/Answer Model [14]). Bob can include his location in the 200 OK response, but this shouldn't be expected due to user timing. If Bob wants to provide his location to Alice after the 200 OK, but before a BYE, the UPDATE Method [9] should be used. Alice's UA replies with an ACK and the session is set up.

Figure 1. does not include any Proxies because in it assumed they would not affect the session set-up with respect to whether or not Alice's location is in a message body part, and Proxies don't react to S/MIME bodies, making their inclusion more or less moot and more complex than necessary.

The most relevant message in Figure 1 having to do with location is (obviously) the message with the location object in it [M1]. So to cut down on length of this document, only the INVITE message in this example will be shown. Section 8.1.1 will give an example of this well formed INVITE message using a Coordinate location format. Section 8.1.2 will give an example of this well formed INVITE message using the civic location format.

8.1.1 UA-to-UA INVITE with Coordinate Location Using S/MIME

Below is a well-formed SIP INVITE Method message to the example in Figure 1 in section 8.1.

[Message 1 in Figure 1]

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
 ;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.example.com
CSeq: 314159 INVITE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; name=smime.p7m
Content-Disposition: attachment;
   filename=smime.p7m  handling=required
Content-Type: multipart/mixed; boundary=boundary1
--boundary1
```



```
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
    <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                 xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
           <gml:location>
             <gml:Point gml:id="point96" srsName="epsg:4326">
               <gml:coordinates>41.87891N
                                87.63649W</gml:coordinates>
             </gml:Point>
           </gml:location>
           <method>dhcp</method>
         </gp:location-info>
         <gp:usage-rules>
           <gp:retransmission-allowed>no</gp:retransmission-allowed>
           <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                           expiry>
         </gp:usage-rules>
       </gp:geopriv>
      </status>
     </tuple>
    </presence>
--boundary1--
```

8.1.1.1 UA-to-UA INVITE with Coordinate Location Not Using S/MIME

Below is a well-formed SIP INVITE Method message to the example in Figure 1 in section 8.1. This message is here to show that although the requirements are mandatory to implement proper security, it is not mandatory to use. This message below is show for those cases where hop-by-hop security is deployed.

```
   [Message 1 in Figure 1]
   INVITE sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/TCP pc33.atlanta.example.com
     ;branch=z9hG4bK74bf9
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 31862 INVITE
   Contact: <sip:alice@atlanta.example.com>
   Content-Type: multipart/mixed; boundary=boundary1
   Content-Length: ...
   --boundary1
   Content-Type: application/sdp
   v=0
   o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
   c=IN IP4 10.1.3.33
   t=0 0
   m=audio 49172 RTP/AVP 0 4 8
   a=rtpmap:0 PCMU/8000
   --broundary1
   Content-Type: application/cpim-pidf+xml
   <?xml version="1.0" encoding="UTF-8"?>
       <presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="urn:opengis:specification:gml:schema-
          xsd:feature:v3.0"
          entity="pres:alice@atlanta.example.com">
        <tuple id="sg89ae">
         <timestamp>2004-07-11T08:57:29Z</timestamp>
         <status>
          <gp:geopriv>
            <gp:location-info>
              <gml:location>
                <gml:Point gml:id="point96" srsName="epsg:4326">
                  <gml:coordinates>41.87891N
                                87.63649W</gml:coordinates>
                </gml:Point>
               </gml:location>
              <method>dhcp</method>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                    expiry>
            </gp:usage-rules>
          </gp:geopriv>
```



```
         </status>
        </tuple>
       </presence>
   --boundary1--
```

8.1.2 UA-to-UA INVITE with Civic Location Using S/MIME

Below is a well-formed SIP INVITE Method message to the example in Figure 1 in section 8.1 using the civic location format.

```
   [Message 1 in Figure 1]
   INVITE sips:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/TLS pc33.atlanta.example.com
    ;branch=z9hG4bK776asdhds
   Max-Forwards: 70
   To: Bob <sips:bob@biloxi.example.com>
   From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
   Call-ID: a84b4c76e66710@pc33.atlanta.example.com
   CSeq: 314159 INVITE
   Contact: <sips:alice@pc33.atlanta.example.com>
   Content-Type: application/pkcs7-mime;
      smime-type=enveloped-data; name=smime.p7m
   Content-Disposition: attachment;
      filename=smime.p7m  handling=required
   Content-Type: multipart/mixed; boundary=boundary1
   --boundary1
   Content-Type: application/sdp
   v=0
   o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
   c=IN IP4 10.1.3.33
   t=0 0
   m=audio 49172 RTP/AVP 0 4 8
   a=rtpmap:0 PCMU/8000
   --boundary1
   Content-type: application/cpim-pidf+xml
   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="urn:opengis:specification:gml:schema-
                    xsd:feature:v3.0"
          entity="pres:alice@atlanta.example.com">
        <tuple id="sg89ae">
         <timestamp>2004-07-11T08:57:29Z</timestamp>
```

```
         <status>
          <gp:geopriv>
            <gp:location-info>
              <cl:civilAddress>
                <cl:country>US</cl:country>
                <cl:A1>Illinois</cl:A1>
                <cl:A3>Chicago</cl:A3>
                <cl:HNO>233</cl:HNO>
                <cl:PRD>South</cl:PRD>
                <cl:A6>Wacker</cl:A6>
                <cl:STS>Drive</cl:STS>
                <cl:PC>60606</cl:PC>
                <cl:LMK>Sears Tower</cl:LMK>
                <cl:FLR>1</cl:FLR>
              <cl:civilAddress>
              <method>dhcp</method>
              <provided-by><nena>www.cisco.com</nena></provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                           expiry>
            </gp:usage-rules>
          </gp:geopriv>
         </status>
        </tuple>
       </presence>
--boundary1--
```

8.1.2.1 UA-to-UA INVITE with Civic Location Not Using S/MIME

Below is a well-formed SIP INVITE Method message to the example in Figure 1 in section 8.1. This message is here to show that although the requirements are mandatory to implement proper security, it is not mandatory to use. This message below is show for those cases where the sending user does not wish to use security mechanisms in transmitting their coordinate location.

```
[Message 1 in Figure 1]
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP pc33.atlanta.example.com
  ;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 31862 INVITE
Contact: <sip:alice@atlanta.example.com>
```



```
Content-Type: multipart/mixed; boundary=boundary1
Content-Length: ...
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--broundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
           <cl:civilAddress>
             <cl:country>US</cl:country>
             <cl:A1>Illinois</cl:A1>
             <cl:A3>Chicago</cl:A3>
             <cl:HNO>233</cl:HNO>
             <cl:PRD>South</cl:PRD>
             <cl:A6>Wacker</cl:A6>
             <cl:STS>Drive</cl:STS>
             <cl:PC>60606</cl:PC>
             <cl:LMK>Sears Tower</cl:LMK>
             <cl:FLR>1</cl:FLR>
           <cl:civilAddress>
           <method>dhcp</method>
           <provided-by><nena>www.cisco.com</nena></provided-by/>
         </gp:location-info>
         <gp:usage-rules>
           <gp:retransmission-allowed>no</gp:retransmission-allowed>
           <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                        expiry>
         </gp:usage-rules>
       </gp:geopriv>
      </status>
     </tuple>
    </presence>
--boundary1--
```

8.1.3 UA-to-UA Location Conveyance Involving 3 Users

In the following example, Alice presents her location in the INVITE to Bob, which Bob 200 OKs with his location as well. Bob then directs Alice to contact Carol with both their locations in the same message. The REFER Method [15] is used in the message sequence, but it does not carry anyone's location within the REFER message. This example is here to show a 3-way communication of location, coupled with how a UA can include someone else's location. This has security implications due to neither primary party in the last location transfer being the owner of the location information. Alice (in this case) MUST adhere to the retention and distribution privacy requirements within Bob's location object regarding his location information prior to considering its inclusion in the INVITE to Carol.

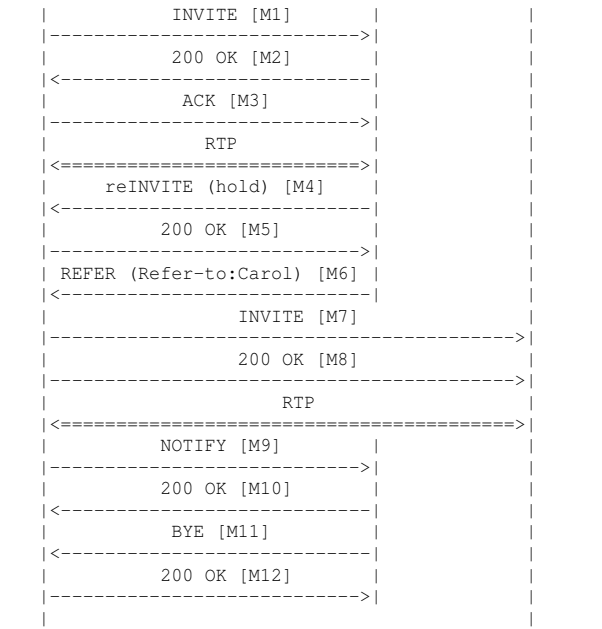


Figure 1a. UA-to-UA with Location in REFER

8.1.3.1 UA-to-UA REFER with Civic Location Using S/MIME

In Figure 1a., we have an example message flow involving the REFER Method. The REFER itself does not carry location objects. We are not including all the messages for space reasons. M1 is a well-formed SIP message that contains Alice's location. M2 is Bob's 200 OK in response to Alice's INVITE, and it contains Bob's Location.

[M1 of Figure 1a] - Alice at Sears Tower

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
 ;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.example.com
CSeq: 314159 INVITE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; name=smime.p7m
Content-Disposition: attachment;
   filename=smime.p7m  handling=required
Content-Type: multipart/mixed; boundary=boundary1
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
```

```
              <cl:civilAddress>
                <cl:country>US</cl:country>
                <cl:A1>Illinois</cl:A1>
                <cl:A3>Chicago</cl:A3>
                <cl:HNO>233</cl:HNO>
                <cl:PRD>South</cl:PRD>
                <cl:A6>Wacker</cl:A6>
                <cl:STS>Drive</cl:STS>
                <cl:PC>60606</cl:PC>
                <cl:LMK>Sears Tower</cl:LMK>
                <cl:FLR>1</cl:FLR>
              <cl:civilAddress>
              <method>dhcp</method>
              <provided-by><nena>www.cisco.com</nena></provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                            expiry>
            </gp:usage-rules>
          </gp:geopriv>
        </status>
       </tuple>
      </presence>
   --boundary1--
```

Bob replies to Alice's INVITE with a 200 OK and includes his location.

```
[M2 of Figure 4] - Bob watching Cubs Game at Wrigley Field
   SIP/2.0 200 OK
   Via: SIP/2.0/TCP pc33.atlanta.example.com
     ;branch=z9hG4bKnashds8 ;received=10.1.3.33
   To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
   From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
   Call-ID: a84b4c76e66710
   CSeq: 314159 INVITE
   Contact: <sip:bob@192.168.10.20>
   Content-Type: application/pkcs7-mime;
      smime-type=enveloped-data; name=smime.p7m
   Content-Disposition: attachment;
      filename=smime.p7m  handling=required
   Content-Type: multipart/mixed; boundary=boundary1
   --boundary1
   Content-Type: application/sdp
   v=0
```



```
   o=bob 2890844530 2890844530 IN IP4 biloxi.example.com
   c=IN IP4 192.168.10.20
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   --boundary1
   Content-type: application/cpim-pidf+xml
   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="urn:opengis:specification:gml:schema-
                     xsd:feature:v3.0"
          entity="pres:bob@biloxi.example.com">
        <tuple id="sg89ae">
         <timestamp>2004-08-6T02:30:29Z</timestamp>
         <status>
          <gp:geopriv>
            <gp:location-info>
              <cl:civilAddress>
                <cl:country>US</cl:country>
                <cl:A1>Illinois</cl:A1>
                <cl:A3>Chicago</cl:A3>
                <cl:A6>Addison</cl:A6>
                <cl:HNO>1060</cl:HNO>
                <cl:PRD>W</cl:PRD>
                <cl:STS>street</cl:STS>
                <cl:LMK>Wrigley Field</cl:LMK>
                <cl:PC>60613</cl:PC>
              <cl:civilAddress>
              <method>dhcp</method>
              <provided-by>www.cisco.com</provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-08-6T18:30:29Z</gp:retention-
                            expiry>
            </gp:usage-rules>
          </gp:geopriv>
         </status>
        </tuple>
       </presence>
   --boundary1--
```

Bob REFERs Alice to Carol, and in M7, Alice includes both locations in a single SIP message. This is possible because Bob set his retention value to "yes", thus allowing Alice to pass his location on to Carol.

```
 [M7 of Figure 1a] - Alice tells Carol where she and Bob are
   INVITE sips:carol@chicago.example.com SIP/2.0
   Via: SIP/2.0/TLS pc33.atlanta.example.com
    ;branch=z9hG4bK776asdhdt
   Max-Forwards: 70
   To: Carol <sips:carol@chicago.example.com>
   From: Alice <sips:alice@atlanta.example.com>;tag=1928301775
   Call-ID: a84b4c76e66711@pc33.atlanta.example.com
   CSeq: 314160 INVITE
   Contact: <sips:alice@pc33.atlanta.example.com>
   Content-Type: application/pkcs7-mime;
      smime-type=enveloped-data; name=smime.p7m
   Content-Disposition: attachment;
      filename=smime.p7m  handling=required
   Content-Type: multipart/mixed; boundary=boundary1
   --boundary1
   Content-Type: application/sdp
   v=0
   o=alice 2890844531 2890844531 IN IP4 atlanta.example.com
   c=IN IP4 10.1.3.33
   t=0 0
   m=audio 49173 RTP/AVP 0 4 8
   a=rtpmap:0 PCMU/8000
   --boundary1
   Content-type: application/cpim-pidf+xml
   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="urn:opengis:specification:gml:schema-
                     xsd:feature:v3.0"
          entity="pres:bob@biloxi.example.com">
        <tuple id="sg89af">
         <timestamp>2004-08-5T02:30:29Z</timestamp>
         <status>
          <gp:geopriv>
            <gp:location-info>
              <cl:civilAddress>
                <cl:country>US</cl:country>
                <cl:A1>Illinois</cl:A1>
                <cl:A3>Chicago</cl:A3>
                <cl:A6>Addison</cl:A6>
                <cl:HNO>1060</cl:HNO>
                <cl:PRD>W</cl:PRD>
                <cl:STS>street</cl:STS>
                <cl:LMK>Wrigley Field</cl:LMK>
                <cl:PC>60613</cl:PC>
```



```
              <cl:civilAddress>
              <method>dhcp</method>
              <method>802.11</method>
              <provided-by>www.cisco.com</provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>yes</gp:retransmission-
                                                        allowed>
              <gp:retention-expiry>2004-08-6T18:30:29Z</gp:retention-
                           expiry>
            </gp:usage-rules>
          </gp:geopriv>
         </status>
        </tuple>
       </presence>
   --boundary1
   Content-type: application/cpim-pidf+xml
   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="urn:opengis:specification:gml:schema-
                     xsd:feature:v3.0"
          entity="pres:alice@atlanta.example.com">
        <tuple id="sg89ae">
         <timestamp>2004-08-6T02:30:29Z</timestamp>
         <status>
          <gp:geopriv>
            <gp:location-info>
              <cl:civilAddress>
                <cl:country>US</cl:country>
                <cl:A1>Illinois</cl:A1>
                <cl:A3>Chicago</cl:A3>
                <cl:HNO>233</cl:HNO>
                <cl:PRD>South</cl:PRD>
                <cl:A6>Wacker</cl:A6>
                <cl:STS>Drive</cl:STS>
                <cl:PC>60606</cl:PC>
                <cl:LMK>Sears Tower</cl:LMK>
                <cl:FLR>1</cl:FLR>
              <cl:civilAddress>
              <method>dhcp</method>
              <method>802.11</method>
              <provided-by>www.marconi.com</provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-08-6T18:30:29Z</gp:retention-
                           expiry>
            </gp:usage-rules>
          </gp:geopriv>
```

```
        </status>
       </tuple>
      </presence>
--boundary1--
```

It is an open question of whether there should be a mechanism to request or require the transmission of an LO. The LO is contained in a body, so the usual sip mechanisms do not apply.

8.2 UA-to-UA Using MESSAGE Method

Anytime a user transmits geographic location outside of an INVITE Request to another user, the MESSAGE Method is to be used. This applies even when two users are in an existing dialog. The logic here is as follows:

- a NOTIFY isn't appropriate because there was not a SUBSCRIBE performed and accepted.
- a UPDATE isn't appropriate because it is for the updating of session capabilities and parameters before a dialog is established, but after a dialog request has been sent. If Alice and Bob were in an existing dialog, UPDATE is already outside its window of usage based on [9].

There is one exception to this for UA-to-UA conveyance: if Alice sent her location in an INVITE, but has moved before receiving a 200 OK, her UA may send an UPDATE to Bob with new location information.

NOTE: A similar use for UPDATE is within the UA-to-Proxy Location Conveyance section of this document.

- reINVITE isn't appropriate because it is only used (or only supposed to be used) for changing the capabilities and/or parameters of an existing dialog. Transferring location has nothing in the UA-to-UA conveyance case to do with the actual dialog, so it does not apply here.

This leaves MESSAGE as the only viable Request Method for location conveyance outside of a dialog between two users (Alice and Bob in this case).

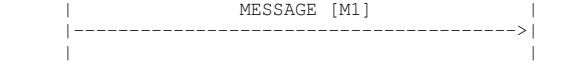


Figure 2. UA-UA with Location in MESSAGE

Section 8.2.1 will give the well formed MESSAGE message containing a well formed Geopriv Location Object using the Coordinate location format that is fully complying with all security requirements - SIPS for hop-by-hop security, and S/MIME for message body confidentiality end-to-end, as well as adhering to the retention and distribution concerns from [7]. Section 8.2.2 will show the Civic Location format alternative to the same location, as conveyed from Alice to Bob. This section does not adhere to confidentiality or integrity concerns of [7], but does convey retention and distribution indicators from Alice.

8.2.1 UA-to-UA MESSAGE with Coordinate Location Using S/MIME

Below is M1 from Figure 2 in section 8.2. that is fully secure and in compliance with Geopriv requirements in [7] for security concerns.

```
[Message 1 in Figure 2]
MESSAGE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
 ;branch=z9hG4bK776asegma
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.example.com
CSeq: 22756 MESSAGE
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; name=smime.p7m
Content-Disposition: attachment;
   filename=smime.p7m  handling=required

Content-Type: multipart/mixed; boundary=boundary1
--boundary1
Content-Type: text/plain
HereÆs my location, Bob?
--broundary1
Content-Type: application/cpim-pidf+xml
Content-Disposition: render
Content-Description: my location
```

```
<?xml version="1.0" encoding="UTF-8"?>
    <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
       xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
           <gml:location>
             <gml:Point gml:id="point96" srsName="epsg:4326">
               <gml:coordinates>41.87891N
                              87.63649W</gml:coordinates>
             </gml:Point>
            </gml:location>
           <method>dhcp</method>
         </gp:location-info>
         <gp:usage-rules>
           <gp:retransmission-allowed>no</gp:retransmission-allowed>
           <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                 expiry>
         </gp:usage-rules>
       </gp:geopriv>
      </status>
     </tuple>
    </presence>
--boundary1--
```

8.2.2 UA-to-UA MESSAGE with Civic Location Not Using S/MIME

Below is a well-formed SIP MESSAGE Method message to the example in Figure 2 in section 8.2 when hop-by-hop security mechanisms are deployed.

```
[Message 1 in Figure 2]
MESSAGE sip:bob@biloxi.example.com SIP/2.0
From: <sip:alice@atlanta.example.com>;tag=34589882
To: <sip:bob@biloxi.example.com>
Call-ID: 9242892442211117@atlanta.example.com
CSeq: 6187 MESSAGE
Content-Type: application/cpim-pidf+xml
Content-ID: <766534765937@atlanta.example.com>
Content-Disposition: render
Content-Description: my location
<?xml version="1.0" encoding="UTF-8"?>
```



```
    <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
       xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
      <tuple id="sg89ae">
       <timestamp>2004-07-11T08:57:29Z</timestamp>
       <status>
        <gp:geopriv>
          <gp:location-info>
            <cl:civilAddress>
              <cl:country>US</cl:country>
              <cl:A1>Illinois</cl:A1>
              <cl:A3>Chicago</cl:A3>
              <cl:HNO>233</cl:HNO>
              <cl:PRD>South</cl:PRD>
              <cl:A6>Wacker</cl:A6>
              <cl:STS>Drive</cl:STS>
              <cl:PC>60606</cl:PC>
              <cl:LMK>Sears Tower</cl:LMK>
              <cl:FLR>1</cl:FLR>
            <cl:civilAddress>
            <method>dhcp</method>
          </gp:location-info>
          <gp:usage-rules>
            <gp:retransmission-allowed>no</gp:retransmission-allowed>
            <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                  expiry>
          </gp:usage-rules>
        </gp:geopriv>
       </status>
      </tuple>
     </presence>
```

8.3 UA-to-UA Location Conveyance Using UPDATE

UPDATE MUST NOT be used to send geographic location from UA-to-UA unless location has already been sent in an INVITE that was the first message in the same dialog set-up. The same security properties used in the INVITE MUST be used in the UPDATE message. The only reason for sending location in an UPDATE message is if Alice's UA (in the common example used throughout this document) has moved prior to receiving Bob's 200 OK to the original INVITE. How this movement is determined is outside the scope of this document, but ultimately should be configurable by local administration or the user of the UA. By how much Alice has moved to trigger the "sense of movement" (i.e. the need to send new location) to Bob is outside the scope of this document, but ultimately should be configurable by local administration or the user of the UA.

In Figure 3., we have an example message flow involving the UPDATE

Method. We are not including all the messages for space reasons. M1 is a well formed SIP message that contains Alice's location. During the session set-up, Alice's UA knows it has moved while knowing too the session has not been formally accepted by Bob. Alice's UA decides to update Bob with her new location with an UPDATE Method message. Messages M2, M3 and M4 have nothing to do with location conveyance, therefore will not be shown in detail. Only M1 and M5 will be shown.

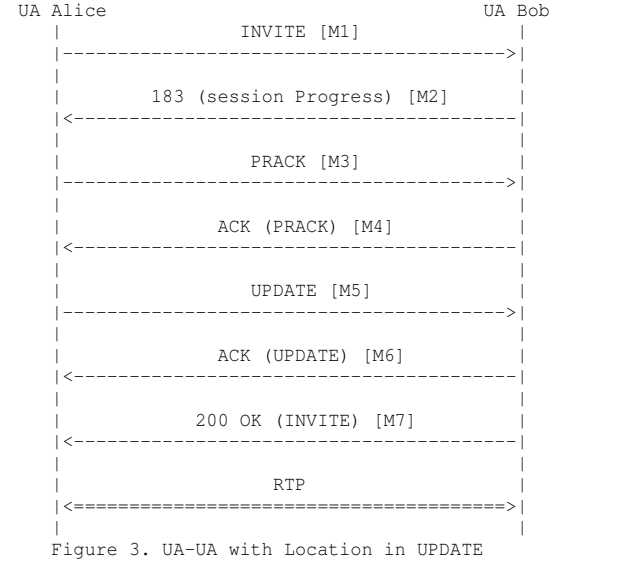


Figure 3. UA-UA with Location in UPDATE

The following section will include the M1 and M5 messages in detail, but only in the civic format.

8.3.1 UA-to-UA UPDATE with Civic Location Not Using S/MIME

Here is the initial INVITE from Alice to Bob.


```
[M1 INVITE to Bob]
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
 ;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.example.com
CSeq: 314159 INVITE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; name=smime.p7m
Content-Disposition: attachment;
   filename=smime.p7m  handling=required
Content-Type: multipart/mixed; boundary=boundary1
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
           <cl:civilAddress>
             <cl:country>US</cl:country>
             <cl:A1>Illinois</cl:A1>
             <cl:A3>Chicago</cl:A3>
             <cl:HNO>233</cl:HNO>
             <cl:PRD>South</cl:PRD>
             <cl:A6>Wacker</cl:A6>
             <cl:STS>Drive</cl:STS>
             <cl:PC>60606</cl:PC>
             <cl:LMK>Sears Tower</cl:LMK>
```

```
                <cl:FLR>1</cl:FLR>
              <cl:civilAddress>
              <method>dhcp</method>
              <method>802.11</method>
             <provided-by>www.cisco.com</provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                           expiry>
            </gp:usage-rules>
          </gp:geopriv>
         </status>
        </tuple>
       </presence>
--boundary1--
```

Alice moves locations (with her UA detecting the movement), causing her UA to generate an UPDATE message ([M5] of Figure 3) prior to her UA receiving a final response from Bob. Here is that message:

M5 UPDATE to Bob

```
UPDATE sips:bob@biloxi.example.com/TCP SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
 ;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928
Call-ID: a84b4c76e66710@pc33.atlanta.example.com
CSeq: 10197 UPDATE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; name=smime.p7m
Content-Disposition: attachment;
   filename=smime.p7m  handling=required
Content-Type: multipart/mixed; boundary=boundary1
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
```



```
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
           <cl:civilAddress>
             <cl:country>US</cl:country>
             <cl:A1>Illinois</cl:A1>
             <cl:A3>Chicago</cl:A3>
             <cl:HNO>250</cl:HNO>
             <cl:PRD>South Upper</cl:PRD>
             <cl:A6>Wacker</cl:A6>
             <cl:STS>Drive</cl:STS>
             <cl:PC>60606</cl:PC>
             <cl:NAM>Venice Cafe</cl:NAM>
             <cl:FLR>1</cl:FLR>
           <cl:civilAddress>
           <method>dhcp</method>
           <method>802.11</method>
           <provided-by>www.t-mobile.com</provided-by/>
         </gp:location-info>
         <gp:usage-rules>
           <gp:retransmission-allowed>no</gp:retransmission-allowed>
           <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                        expiry>
         </gp:usage-rules>
       </gp:geopriv>
      </status>
     </tuple>
    </presence>
--boundary1--
```

8.4 UA-to-UA Location Conveyance Using PUBLISH

** This section could not be completed before submission time and will be completed shortly after IETF60 (unless). A thousand pardons.

9. Special Considerations for Emergency Calls

When a Proxy Server knows to look for the location message body to route an emergency call as in [11].

Emergency calls, which might be detected as detailed in 3, have special rules for conveyance of location:

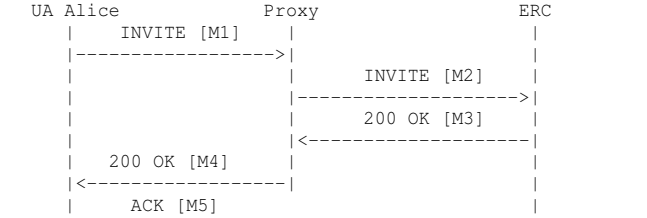
1. An emergency call MUST have all LI available to the UA, if any, sent with the INVITE, and subsequent UPDATE or reINVITE messages as a PIDF-LO in a body
2. The LO must be protected with sips: UNLESS the attempt to establish hop-by-hop TLS connections fails and cannot reasonably be established in a very short (less than a second) time. In such a case, the LO SHOULD be sent without TLS ONLY for those hops that cannot support TLS establishment.
3. User Agents MUST NOT use S/MIME

Proxies MUST NOT remove a location message body at any time. If there is a condition that a Proxy adds a location message body, it:

4. MUST NOT produce a message length over current SIP message limits (1300 bytes [per 3428])
5. MUST indicate within that added message body that body came from that server (by some naming convention not defined here)

9.1 UA-to-Proxy Routing the Message with INVITE (secure)

When Alice signifies "sos@" [per 3], her UA must understand this message MUST NOT use S/MIME for the message body, because this is an emergency call - otherwise the message will not properly route to the correct destination. Two definite possibilities will exist for how this message flow will occur [note: the message flows are not being defined here, they are defined in [11], but two are shown here to show the messages themselves]. The first possibility has Alice sending her INVITE to her first hop Proxy, which recognizes the message as an emergency message. The Proxy knows to look into the message bodies for the location body; determine where Alice is and route the call to the appropriate ERC. This is shown in Figure 4A.



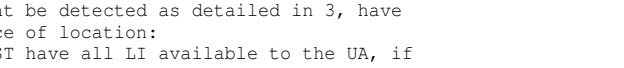


Figure 4A. UA-PROXY with Location in INVITE

```
[M1 of  Figure 4A]
INVITE sips:sos@atlanta.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
  ;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sips:alice@atlanta.example.com>;tag=9fxced76sl
To: <sips:sos@atlanta.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 31862 INVITE
Contact: <sips:alice@atlanta.example.com>
Content-Type: multipart/mixed; boundary=boundary1
Content-Length: ...
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
```

Once the Proxy receives M1 and recognizes it as an emergency INVITE Request, this proxy knows to look into the message body for a location body part to determine the location of the UAC in order to match the location to an ERC. Once this look-up occurs, the message is sent directly to the ERC (in message [M2]).

```
[M2 of Figure 4A] - Proxy has determined when to send message
INVITE sips:sos@192.168.10.20 SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
  ;branch=z9hG4bK74bf9
Max-Forwards: 69
From: Alice <sips:alice@atlanta.example.com>;tag=9fxced76sl
To: <sips:sos@atlanta.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 31862 INVITE
Contact: <sips:alice@atlanta.example.com>
```

```
   Content-Type: multipart/mixed; boundary=boundary1
   Content-Length: ...
   --boundary1
   Content-Type: application/sdp
   v=0
   o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
   c=IN IP4 10.1.3.33
   t=0 0
   m=audio 49172 RTP/AVP 0 4 8
   a=rtpmap:0 PCMU/8000
   --boundary1
   Content-type: application/cpim-pidf+xml
   <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gml="urn:opengis:specification:gml:schema-
                     xsd:feature:v3.0"
          entity="pres:alice@atlanta.example.com">
        <tuple id="sg89ae">
         <timestamp>2004-07-11T08:57:29Z</timestamp>
         <status>
          <gp:geopriv>
            <gp:location-info>
              <cl:civilAddress>
                <cl:country>US</cl:country>
                <cl:A1>Illinois</cl:A1>
                <cl:A3>Chicago</cl:A3>
                <cl:HNO>233</cl:HNO>
                <cl:PRD>South</cl:PRD>
                <cl:A6>Wacker</cl:A6>
                <cl:STS>Drive</cl:STS>
                <cl:PC>60606</cl:PC>
                <cl:LMK>Sears Tower</cl:LMK>
                <cl:FLR>1</cl:FLR>
              <cl:civilAddress>
              <method>dhcp</method>
              <method>802.11</method>
              <provided-by>www.t-mobile.com</provided-by/>
            </gp:location-info>
            <gp:usage-rules>
              <gp:retransmission-allowed>no</gp:retransmission-allowed>
              <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                           expiry>
            </gp:usage-rules>
          </gp:geopriv>
         </status>
        </tuple>
       </presence>
```



```
--boundary1--
```

The second probability in message flows is in Figure 4B. in which the first hop Proxy does not either: understand location, or does not know where the appropriate ERC is to route the message to. In either case, that Proxy forwards the message to another Proxy for proper message routing ([11] talks to how this occurs).

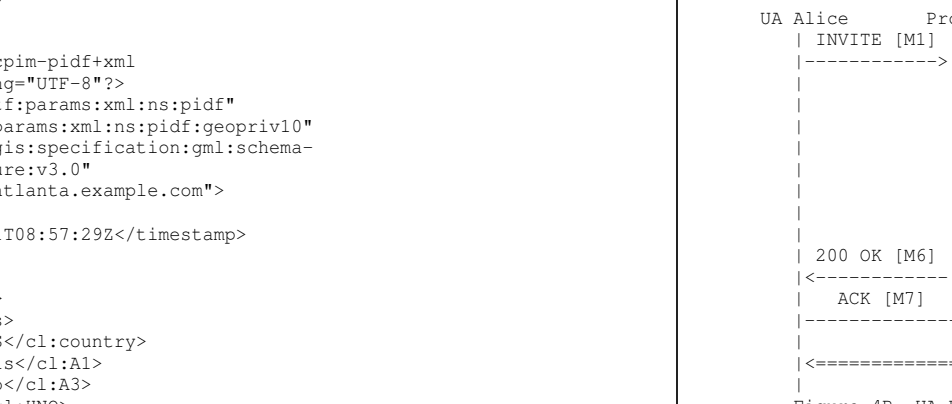


Figure 4B. UA-PROXY with Location in INVITE

In message flows similar to 4A and/or 4B, the Record-Route header could be added by the proxies, this is OPTIONAL in usage and left to other documents to refine.

In the case of an identifiable emergency call, something that cannot happen is for any Proxy to Challenge [per 1] the INVITE message. In fact, while usage of the SIPS URI is encouraged and SHOULD be used, it MUST NOT be mandatory for successful message routing. If the first SIPS INVITE fails for security property reasons, the second attempt by Alice (in these examples) MUST be allowed to be in the clear, not challenged, and routed properly. Security mechanisms MUST NOT fail any call attempt, and if they do once, they MUST NOT be mandatory for the subsequent attempt for a successful session set-up to an ERC. The results of this are that the Proxy that failed the first attempt for security reasons MUST be aware of this failed attempt for the subsequent attempt that MUST process without failure a second time. It must be assumed that the INVITE in any instance is considered "well formed".

The remaining messages in both 4A and 4B are not included at this time. If the working groups wants these added, they will be in the next revision of this document.

9.1.1 UA-to-Proxy Routing the Message with INVITE (unsecure)

Below can be considered the initial unsecure INVITE M1 from Figures 4A and 4A, or the second attempt message to an initial message that was failed by a Proxy. This version of M1 is not using any security measures and is using the civic format message body that is the identical location to the previous example.

```
[Message M1 from Figure 4A]
INVITE sip:sos@atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP pc33.atlanta.example.com
  ;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: <sip:sos@atlanta.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 31862 INVITE
Contact: <sip:alice@atlanta.example.com>
Content-Type: multipart/mixed; boundary=boundary1
Contact-Length: ...
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
      <tuple id="sg89ae">
       <timestamp>2004-07-11T08:57:29Z</timestamp>
       <status>
        <gp:geopriv>
          <gp:location-info>
```



```
            <cl:civilAddress>
              <cl:country>US</cl:country>
              <cl:A1>Illinois</cl:A1>
              <cl:A3>Chicago</cl:A3>
              <cl:HNO>233</cl:HNO>
              <cl:PRD>South</cl:PRD>
              <cl:A6>Wacker</cl:A6>
              <cl:STS>Drive</cl:STS>
              <cl:PC>60606</cl:PC>
              <cl:LMK>Sears Tower</cl:LMK>
              <cl:FLR>1</cl:FLR>
            <cl:civilAddress>
            <method>dhcp</method>
            <method>802.11</method>
            <provided-by>www.t-mobile.com</provided-by/>
          </gp:location-info>
          <gp:usage-rules>
            <gp:retransmission-allowed>no</gp:retransmission-allowed>
            <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                        expiry>
          </gp:usage-rules>
        </gp:geopriv>
       </status>
      </tuple>
     </presence>
--boundary1--
```

9.2 UA-to-Proxy Routing with UPDATE

If the previous example of the location contained in the INVITE were to account for the movement of Alice (and her UA) before the ERC responded with a 200 OK, the UPDATE method is the appropriate SIP Request Method to use to update the proxies and ERC personnel that Alice has moved geo-locations from where she initially made her set-up request.

In this scenario (shown in the call flow of Figure 5A), Alice sending the UPDATE message here may cause the Proxy to CANCEL an existing pending INVITE Request, and retransmit INVITE to a NEW ERC(2), for example, if she walked across a street into a new ERC coverage area. The Proxy MUST remain transaction stateful in order to be aware of the 200 OK Response from ERC1. Upon receiving the UPDATE from Alice and analyzing the location provided by the message looking for a geographic change, either forwarding that message to ERC1 if the change is still within ERC1's coverage area, or deciding to forward a message to another ERC covering where Alice is now (ERC2 in this case) with her new location. If the later change in destinations is required, the Proxy MUST CANCEL the pending INVITE to ERC1 (with a 487 "terminated request" being the specified response).

SIPS MUST be used by Alice initially. Upon any failure of the initial Request, Alice's UA can decide to send the new message without SIPS.

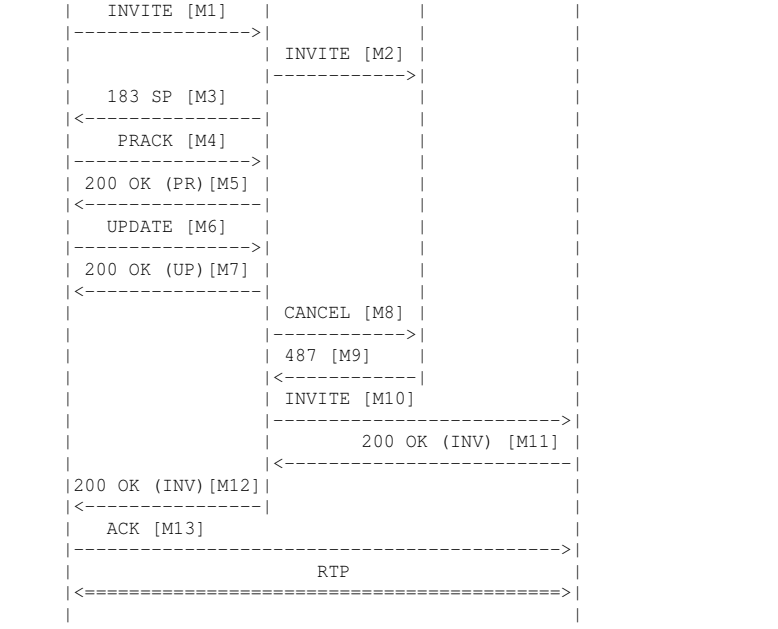


Figure 5A. UA-PROXY with Location in UPDATE

** see new open issue #9 for the problems with messages 8 through 10
** of the above flow.

9.2.1 UA-to-Proxy Routing the Message with UPDATE (secure)

```
INVITE sip:sos@atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP pc33.atlanta.example.com
  ;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: <sip:sos@atlanta.example.com>
```



```
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 31862 INVITE
Contact: <sip:alice@atlanta.example.com>
Content-Type: multipart/mixed; boundary=boundary1
Contact-Length: ...
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
      <tuple id="sg89ae">
       <timestamp>2004-07-11T08:57:29Z</timestamp>
       <status>
        <gp:geopriv>
          <gp:location-info>
            <cl:civilAddress>
              <cl:country>US</cl:country>
              <cl:A1>Illinois</cl:A1>
              <cl:A3>Chicago</cl:A3>
              <cl:HNO>233</cl:HNO>
              <cl:PRD>South</cl:PRD>
              <cl:A6>Wacker</cl:A6>
              <cl:STS>Drive</cl:STS>
              <cl:PC>60606</cl:PC>
              <cl:LMK>Sears Tower</cl:LMK>
              <cl:FLR>1</cl:FLR>
            <cl:civilAddress>
            <method>dhcp</method>
            <method>802.11</method>
           <provided-by>www.cisco.com</provided-by/>
          </gp:location-info>
          <gp:usage-rules>
            <gp:retransmission-allowed>no</gp:retransmission-allowed>
            <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                          expiry>
          </gp:usage-rules>
        </gp:geopriv>
```

```
         </status>
        </tuple>
       </presence>
--boundary1--
```

Alice moves locations (with her UA detecting the movement), causing her UA to generate an UPDATE message ([M5] of Figure 3) prior to her UA receiving a final response from the ERC. In this case, Alice has walked across the South Wacker Drive to another building. Here is that message:

[M5 UPDATE to ERC]

```
UPDATE sips:bob@biloxi.example.com/TCP SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com
 ;branch=z9hG4bK776asdhds
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: <sip:sos@atlanta.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 10187 UPDATE
Contact: <sip:alice@atlanta.example.com>
Content-Type: multipart/mixed; boundary=boundary1
Contact-Length: ...
--boundary1
Content-Type: application/sdp
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
c=IN IP4 10.1.3.33
t=0 0
m=audio 49172 RTP/AVP 0 4 8
a=rtpmap:0 PCMU/8000
--boundary1
Content-type: application/cpim-pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gml="urn:opengis:specification:gml:schema-
                  xsd:feature:v3.0"
       entity="pres:alice@atlanta.example.com">
     <tuple id="sg89ae">
      <timestamp>2004-07-11T08:57:29Z</timestamp>
      <status>
       <gp:geopriv>
         <gp:location-info>
           <cl:civilAddress>
             <cl:country>US</cl:country>
```



```
             <cl:A1>Illinois</cl:A1>
             <cl:A3>Chicago</cl:A3>
             <cl:HNO>250</cl:HNO>
             <cl:PRD>South Upper</cl:PRD>
             <cl:A6>Wacker</cl:A6>
             <cl:STS>Drive</cl:STS>
             <cl:PC>60606</cl:PC>
             <cl:NAM>Venice Cafe</cl:NAM>
             <cl:FLR>1</cl:FLR>
           <cl:civilAddress>
           <method>dhcp</method>
           <method>802.11</method>
           <provided-by>www.t-mobile.com</provided-by/>
         </gp:location-info>
         <gp:usage-rules>
           <gp:retransmission-allowed>no</gp:retransmission-allowed>
           <gp:retention-expiry>2004-07-13T14:57:29Z</gp:retention-
                        expiry>
         </gp:usage-rules>
       </gp:geopriv>
      </status>
     </tuple>
    </presence>
--boundary1--
```

9.2.2 UA-to-Proxy Routing the Message with UPDATE (unsecure)

left blank for now

10. Meeting RFC3693 Requirements

Section 7.2 of [7] details the requirements of a "using protocol". They are:

Req. 4. The using protocol has to obey the privacy and security instructions coded in the Location Object and in the corresponding Rules regarding the transmission and storage of the LO.

This document requires, in Section 7, that SIP entities sending or receiving location MUST obey such instructions.

Req. 5. The using protocol will typically facilitate that the keys associated with the credentials are transported to the respective parties, that is, key establishment is the responsibility of the using protocol.

[1] and the documents it references define the key establish mechanisms.

Req. 6. (Single Message Transfer) In particular, for tracking of small target devices, the design should allow a single message/packet transmission of location as a complete transaction.
This document specifies that the LO be contained in the body of a single message.

11. Current Known Open issues

This is a list of open issues that have not yet been addressed to conclusion:

1) Whether SIP Proxies SHOULD be able to insert location information into an emergency call set-up (the INVITE)?
   1a) This has the additional implication of whether or not, or regardless of the fact the UAC already inserted location into the sos@homedomain INVITE.
   1b) Should the Proxy somehow differentiate its location information from that provided by the UAC (with each LI having a SIP entity (type?) originator label?
   1c) Should there be any behavior difference with respect to Open Issue #1b if the Proxy does not know or cannot tell if the UAC inserted location information (further emphasizing the need for some form of originator label)?
2) Whether SIP Proxies SHOULD be able to return location information in a Redirect message to the UAC making the emergency call?

12. New Open Issues

These are new open issues to be addressed within this document or the topics/areas dropped from consideration:

1) How do we handle proxy authenticated location?
2) What do we do in an Offer/Answer model where the INV contains an Offer to the UAS asking which format they want to receive?
3) What do we do with Alice wanting Bob's Location in the 200 OK (to her INVITE with location)?
4) What about a new 4XX error for unknown or bad location given?
5) There is the case in the Proxy Routing in which a UAC sent an INVITE to sos@ without a location message body. Does this

necessitate the need for a 4XX level error informing the UAC to "retry with the location message body included this time"? Another spin on this is if the UAC doesn't know it's location and wants to ask a Proxy server to include the UAC's location if it is known to the Proxy...

6) How or should we get into a Redirect message from a PS that contains a Location body for that UAC? Should we RECOMMEND a UAC that receives a 3XX Reply to an INVITE that contains a Location body with a presence line signifying the UAC, the UAC MUST include that Location body in the new INVITE?
   6a) What if the UAC already sent a Location body in the original message, should it replace the location body with what the PS included, or include both?
      6ai) If we state "both", which we agreed in the past is a good idea, I see no way in the PIDF-LO or in MIME to denote which message body came from Alice and which came from the PS...
7) The authors failed to get this document reclassified into a specification effort (from a requirements ID effort)
   7a) will re-request to the ADs after IETF60 for this
8) Req U-PS7 (Proxy Assertion of a Location body) is not addressable yet in SIP (as Identity is barely addressable).
   8a) Should this requirement remain as a goal?
9) From section 9.2 (Emergency call with an updated location), if Alice does venture into another coverage area, how does her new UPDATE with new location get sent to a second (and now appropriate) ERC(2)?
   The pending INVITE needs to be cancelled or able to be sequentially forked (which not all Proxies will be able to do). Without that occurring, the new UPDATE will not cause a new INVITE to be originated from the Proxy towards ERC2... and what happens to the UPDATE message (which cannot be an original request into ERC2)?

13. Security Considerations

Conveyance of geo-location of a UAC is problematic for many reasons. This document calls for that conveyance to normally be accomplished through secure message body means (like S/MIME or TLS). In cases where a session set-up is routed based on the location of the UAC initiating the session or SIP MESSAGE, securing the location with an

end-to-end mechanism such as S/MIME is problematic.

14. IANA Considerations

There are no IANA considerations within this document at this time.

15. Acknowledgements

To Dave Oran for helping to shape this idea. To Jon Peterson and Dean Willis on guidance of the effort. To Henning Schulzrinne, Jonathan Rosenberg, Dick Knight, and Keith Drage for constructive feedback.

16. References

16.1 References - Normative

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol ", RFC 3261, June 2002

[2] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Mar. 1997.

[3] H. Schulzrinne, "draft-ietf-sipping-sos-00.txt", Internet Draft, Feb 2004, Work in progress

[4] B. Campbell, Ed., J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging" , RFC 3428, December 2002

[5] J. Polk, J. Schnizlein, M. Linsner, " DHCP Option for Location Configuration Information", RFC 3825, July 2004

[6] H. Schulzrinne, "draft-ietf-geopriv-dhcp-civic-03.txt", Internet Draft, July 04, Work in progress

[7] J. Cuellar, J. Morris, D. Mulligan, J. Peterson. J. Polk, "Geopriv Requirements", RFC 3693, February 2004

[8] J. Rosenberg, "Requirements for Session Policy for the Session Initiation Protocolö, draft-ietf-sipping-session-policy-req-00", Internet Draft, June, 2003, "work in progress"

[9] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002

[10] A. Niemi, Ed., "draft-ietf-sip-publish-04", Internet Draft, May 2004, work in progress

[11] H. Schulzrinne, B. Rosen, "draft-schulzrinne-sipping-emergency-arch", Internet Draft, Feb 2004, work in progress

[12] "Requirements for End to Middle Security in SIP", draft-ietf-sipping-e2m-sec-reqs-03.txt, Internet Draft, June 2004, work in progress,

[13] J. Peterson, "draft-ietf-geopriv-pidf-lo-02", Internet Draft, May 2004, work in progress

[14] J. Rosenberg, H. Schulzrinne, "The Offer/Answer Model with Session Description Protocol", RFC 3264, June 2002

[15] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003

17. Author Information

James M. Polk
Cisco Systems
2200 East President George Bush Turnpike 33.00111N
Richardson, Texas 75082 USA 96.68142W
jmpolk@cisco.com

Brian Rosen
Marconi Communications, Inc.
2000 Marconi Drive 40.65923N
Warrendale, PA 15086 80.09958W
Brian.rosen@marconi.com

Full 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. 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.

Intellectual Property

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 procedures with respect to rights in RFC 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.

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

The Expiration date for this Internet Draft is:
January 19th, 2005

SIPPING Working Group
Internet-Draft
Expires: January 5, 2005

G. Camarillo
Ericsson
A. Niemi
H. Khartabil
M. Isomaki
M. Garcia-Martin
Nokia
July 7, 2004

Refering to Multiple Resources in the Session Initiation Protocol (SIP)

draft-ietf-sipping-multiple-refer-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 5, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document defines extensions to the SIP REFER method so that this method can be used to refer servers to multiple resources. These extensions include the use of pointers to URI-lists in the Refer-To header field and the multiple-refer SIP option-tag.

Table of Contents

1. Introduction

The SIP REFER method [5] allows a user agent to request a server to send a request to a third party. Still, a number of applications need to request a server to initiate transactions towards a set of destinations. In one example, the moderator of a conference may want the conference server to send BYE requests to a group of participants. In another example, the same moderator may want the conference server to INVITE a set of new participants.

We define an extension to REFER so that REFER can be used to refer servers to multiple destinations. In addition, we use the REFER extension defined in [7] which suppresses REFER's implicit subscription.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

We define the following three new terms:

REFER-Issuer: the user agent issuing the REFER request.
REFER-Recipient: the user agent receiving the REFER request.
REFER-Target: the user agent designated in the Refer-To URI.

3. Overview of operation

This document defines an extension to the SIP REFER method [5] that allows a SIP UAC to include a list of REFER-Targets in a REFER request and send it to a server. The server will create a new request for each entry in the list of REFER-Target URIs.

We represent the multiple REFER-Targets of a REFER using the URI-list format specified in [8]. A UAC (User Agent Client) that wants to refer a server to a set of destinations creates a SIP REFER request. The Refer-To header contains a pointer to a URI-list, which is included in a body part, and two option-tags in the Required header field: "multiple-refer" and "norefersub". The former indicates the requirement to support the functionality described in this specification and the latter removes the implicit subscription associated to REFER requests by default.

When the server receives such request it creates a new request per destination and sends them.

This document does not provide any mechanism for UACs to find out about the results of a REFER with multiple REFER-Targets. Furthermore, we do not provide support for the implicit subscription mechanism that is part of the SIP REFER method. The way UACs are kept informed about the results of a REFER is service specific. For example, a UAC sending a REFER to INVITE a set of participants to a conference may discover which participants were successfully brought in into the conference by using the conference package [10]

4. The multiple-refer SIP Option-Tag

We define a new SIP option-tag for the Require and Supported header fields: "multiple-refer".

A user agent including the "multiple-refer" option-tag in a Supported header indicates compliance with this specification.

A user agent generating a REFER with a pointer to a URI-list in its Refer-To header field MUST include the "multiple-refer" option-tag in the Require header field of the REFER.

5. Suppressing REFER's Implicit Subscription

REFER requests with a single REFER-Target establish a subscription implicitly. The REFER-Issuer is informed about the result of the transaction towards the REFER-Target through this implicit subscription.

In the case of a REFER-Issuer that generates a REFER with multiple REFER-targets, the REFER-Issuer is typically already subscribed to other event package that can provide the information about the result of the transactions towards the REFER-Targets. For example, a moderator instructing a conference server to send a BYE request to a set of participants is usually subscribed to the conference state event package for the conference. Notifications to this event package will keep the moderator and the rest of the subscribers informed of the current list of conference participant.

Consequently, we have decided to remove the implicit subscription from a multiple REFER request. So, a SIP REFER-Issuer generating a REFER request with multiple REFER-Targets MUST include the 'norefersub' option-tag in a Require header field to indicate that no notifications about the requests should be sent to the REFER-Issue. The 'norefersub' SIP option-tag is defined in [7] and suppresses the REFER's implicit subscription.

6. Behavior of SIP User Agents

Implementations of this specification MUST suppot the transfer mechanism for URI-lists defined in [8].

6.1 Behavior of SIP REFER-Issuers

As indicated in Section 4 and Section 5 a SIP REFER-Issuer that creates a REFER request with multiple REFER-Targets includes a "multiple-refer" and a "norefersub" option-tags in the Require header field.

The Refer-To header field of a REFER request with multiple REFER-Targets MUST contain a pointer (i.e., a Content-ID URL [2]) that points to the body part (whose disposition type is "uri-list") that carries the URI-list.

As described in [8], the default format for URI-lists in SIP is the XCAP resource list format [6]. Still, specific services need to describe which information clients should include in their URI lists, as described in [8].

SIP REFER-Issuers generating REFERs with multiple REFER-Targets SHOULD use flat lists (i.e., no hierarchical lists), SHOULD NOT use any entry's attributes but "uri", and SHOULD NOT include any elements inside entries but "display-name" elements.

6.2 Behavior of REFER-Recipients

A REFER-Recipient receiving a URI-list with more information than what we have described in Section 6.1 SHOULD discard all the extra information.

The REFER-Recipient follows the rules in Section 2.4.2 of RFC 3515 [5] to determine the status code of the response to the REFER.

7. Example

The following is an example of a REFER request with multiple REFER-Targets. The REFER's Refer-To header field carries a pointer to the message body, which carries a list with the URIs of the REFER-Targets. The REFER's Require header field carries both the "multiple-refer" and the "norefersub" option-tags.


```
REFER sip:conf-123@example.com
SIP/2.0 Via: SIP/2.0/TCP client.chicago.example.com
        ;branch=z9hG4bKhjhs8ass83
Max-Forwards: 70
To: Conference 123 <sip:conf-123@example.com>
From: Carol <sip:carol@chicago.example.com>;tag=32331
Call-ID: d432fa84b4c76e66710
CSeq: 2 REFER
Contact: <sip:carol@client.chicago.example.com>
Refer-To: <cid:cn35t8jf02@example.com>
Require: multiple-refer, norefersub
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Allow-Events: dialog
Accept: application/sdp, message/sipfrag
Content-Type: application/resource-lists+xml
Content-Disposition: uri-list
Content-Length: 307
Content-ID: <cn35t8jf02@example.com>

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com?method=BYE" />
    <entry uri="sip:joe@example.org?method=BYE" />
    <entry uri="sip:ted@example.net?method=BYE" />
  </list>
</resource-lists>
```

Figure 1: REFER request with multiple REFER-Targets

8. Security Considerations

The Security Considerations Section of the Requirements and Framework for SIP URI-List Services [9] discusses issues related to SIP URI-list services. Given that a server accepting REFERs with multiple REFER-targets acts as an URI-list service, implementations of this type of server MUST follow the security-related rules in [9]. These rules include mandatory authentication and authorization of clients, and opt-in lists.

Additionally, servers SHOULD only accept REFER requests within the context of an application the server understands (e.g., a conferencing application). This implies that servers MUST NOT accept REFERs for methods they do not understand. The idea behind these two rules is that servers are not used as dumb servers whose only function is to fan-out random messages they do not understand.

9. IANA Considerations

This document defines a SIP option-tag (multiple-refer) in Section 4. This option-tag should be registered in the SIP parameter registry (http://www.iana.org/assignments/sip-parameters).

SIP user agents that place the multiple-refer option-tag in a Supported header field understand REFER requests with multiple REFER-Targets.

10. References

10.1 Normative References

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

[2] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, August 1998.

[3] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[4] Sparks, R., "Internet Media Type message/sipfrag", RFC 3420, November 2002.

[5] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[6] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[7] Olson, S., "Extended-REFER framework and other REFER extensions", draft-olson-sipping-refer-extensions-01 (work in progress), February 2004.

[8] Camarillo, G., "Providing a Session Initiation Protocol (SIP) Application Server with a List of URIs", draft-camarillo-sipping-uri-list-01 (work in progress), February 2004.

[9] Camarillo, G., "Requirements for Session Initiation Protocol (SIP) Exploder Invocation", draft-camarillo-sipping-exploders-02 (work in progress), February 2004.

10.2 Informational References

[10] Rosenberg, J. and H. Schulzrinne, "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-03 (work in progress), February 2004.

[11] Camarillo, G., "A Transaction Event Package for the Session Initiation Protocol (SIP)", draft-camarillo-sipping-transac-package-00 (work in progress), February 2004.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Aki Niemi
Nokia
P.O. Box 321
NOKIA GROUP, FIN 00045
Finland

EMail: Aki.Niemi@nokia.com

Hisham Khartabil
Nokia
P.O. Box 321
NOKIA GROUP, FIN 00045
Finland

EMail: Hisham.Khartabil@nokia.com

Markus Isomaki
Nokia
Itamerenkatu 11-13
Helsinki 00180
Finland

EMail: Markus.Isomaki@nokia.com

Miguel A. Garcia-Martin
Nokia
P.O.Box 407
NOKIA GROUP, FIN 00045
Finland

EMail: miguel.an.garcia@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.

Session Initiation Proposal Investigation Working Group
Internet-Draft
Expires: January 8, 2005

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson
J. Rosenberg
dynamicsoft

July 10, 2004

Profile Data for Session Initiation Protocol (SIP) Policies

draft-ietf-sipping-session-indep-policy-00

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 8, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This draft specifies an XML schema for profile data for SIP session policies. This schema can be used within the user agent profile devliery framework to implement session-independent SIP session policies.

Table of Contents

1. Introduction

Some domains have policies in place, which impact the sessions established using the Session Initiation Protocol (SIP). These policies are typically needed to support the operation of the network infrastructure or certain services. For example, a SIP user agent might be located in a domain that is behind a Network Address Translator (NAT). This domain might have a policy in place that requires the user agent to contact a TURN [10] relay before setting up a session. Information about this policy is essential for a user agent to successfully set up a session.

In another example, SIP is used in a wireless network. The network provider has limited resources for media traffic. During periods of high activity, the provider would like to restrict codec usage on the network to lower rate codecs. In existing approaches, this is frequently accomplished by having the proxies examine the SDP [2] in the body and remove the higher rate codecs or reject the call and require the UA to start over with a different set of codecs. Having information about the current policy would enable user agents to initiate a session with an acceptable codec.

In a third example, a domain has established policies regarding the type of user agents that can use their network. For example, a domain could require that user agents using its network use a particular protocol (e.g., SIP) with a set of extensions (e.g., preconditions must be used). A user agent needs to know the exact policy of a domain in order to be able to use the right configuration to send and receive traffic in that domain.

Some domains have policies in place that are enforced by network elements. For example, a domain might have a configuration in which all packets containing a certain voice encoding are dropped. Unfortunately, enforcement mechanisms usually do not inform the user about the policies they are enforcing and silently keep the user from doing anything against them. This may lead to the malfunctioning of devices that is in-apprehensible to the user. With session policies, the user could decide to switch to a different codec or connect to a domain with less stringent policies.

Session policies may be specific to a certain session and may change from session to session. Such policies can be set up using the framework for session-specific policies [3]. Other session policies remain in place for a longer period of time, typically in the range of hours or days. In principle, these policies could also be set up on a session-to-session basis. However, establishing the same policies over and over again is expensive, causing the continuous transmission of the same information during session setup, and

possibly adding to session setup latencies. It is therefore desirable to enable user agents to obtain the policies relevant for them and to inform the user agents about changes in these policies.

This draft specifies a XML schema for media and protocol user agent profile data. The media data defines properties of media streams transmitted by a user agent. The protocol data defines the methods, extensions, bodies, etc. that should be supported by a user agent. These formats can be used to define session policy documents. Session policy documents can be transmitted to user agents as part of their device configuration using the Framework for SIP User Agent Profile Delivery [8].

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, [1] and indicate requirement levels for compliant implementations.

3. Considerations for Policy-related Profile Data

Policy documents should support versioning so that the recipients of policy document can properly order them. This may be achieved using a version attribute.

A policy document may contain multiple policies. Each policy in the document may have a different scope. For example, a policy for firewall traversal would only apply to external calls whereas a policy limiting the bandwidth available could be in effect during peak hours. A policy document may define a scope attribute that specifies to which sessions a certain policy applies. Possible scopes are:

- Time and day: limits the use of a policy to certain times or days.
- Local entity: limits the use of a policy to a specific to a certain local user. This is in particular useful for devices that supports multiple identities.
- Remote entity: limits the use of a policy to sessions involving certain remote addresses, for example all non-local addresses.
- Media streams: limits the use of a policy to certain media streams.

The use of policies may be mandatory or optional. A policy document may specify whether a policy is mandatory or optional.

4. User Agent Profile Data for Session Policies

TODO: This specification needs to be alinged the schema for SIP protocol user agent profile data sets so that it can co-exist with other data delivered through the user agent configuration framework.

A session policy document is an XML document that MUST be well-formed and SHOULD be valid. Policy documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying session policy documents. The namespace URI for elements defined by this specification is a URN [5], using the namespace identifier 'ietf' defined by RFC 2648 [6] and extended by [4]. This URN is:

urn:ietf:params:xml:ns:sessionpolicy

A session policy document begins with the root element tag "sessionpolicy".

4.1 Policy Document Format

A session policy document starts with a sessionpolicy element. This element has three mandatory attributes:

version: This attribute allows the recipient of session policy information documents to properly order them. Versions start at 0, and increment by one for each new document sent to a subscriber. Versions are scoped within a subscription. Versions MUST be representable using a 32 bit integer.

domain: This attribute contains the domain the policy belongs to.

entity: This attribute contains a URI that identifies the user whose policy information is reported in the remainder of the document.

The sessionpolicy element has a series of sessionpolicy sub-elements: zero or one protocols element and zero or one media element.

4.1.1 Protocols Element

The protocols element contains a series of protocol sub-elements. Each protocol sub-element contains the policy related to the usage of a particular protocol.

The protocol element has a single mandatory attribute, name. The name attribute identifies a protocol the policy of each protocol element is referring to. The protocol element has a series of sub-elements: methods, option-tags, feature-tags, and bodies.

4.1.1.1 Methods Element

The methods element contains a default-policy attribute and method elements. The default-policy attribute contains the policy for methods that are not listed as method elements. A method element has two attributes: name and policy. The name attribute identifies a method, and the policy attribute contains the policy for that method (allowed or disallowed).

4.1.1.2 Option-tags Element

The option-tags element contains a default-policy attribute and option-tag elements. The default-policy attribute contains the policy for option-tags that are not listed as option-tag elements. An option-tag element has two attributes: name and policy. The name attribute identifies a method, and the policy attribute contains the policy for that method (mandatory, allowed, or disallowed).

4.1.1.3 Feature-tags Element

The feature-tags element contains a default-policy attribute and feature-tag elements. The default-policy attribute contains the policy for feature-tags that are not listed as feature-tag elements. An feature-tag element has two attributes: name and policy. The name attribute identifies a method, and the policy attribute contains the policy for that method (allowed, or disallowed).

4.1.1.4 Bodies Element

The bodies element contains a default-policy attribute, a default-encryption attribute and body-disposition elements. The default-policy attribute contains the policy for body dispositions that are not listed as body-disposition elements. The default-encryption attribute contains the encryption policy for body dispositions that are not listed as body-disposition elements.

A body-disposition element can have a number of attributes: name, policy, default-policy, and encryption. The name attribute identifies a body-disposition, and the policy attribute contains the policy for that body-disposition (allowed, or disallowed). The default-policy attribute contains the policy for body formats that are not listed as body-format elements. The encryption attribute indicates whether or not encryption is allowed for a particular body disposition.

A body-disposition element contains body-format elements. A body-format element can have a two attributes: name and policy. The name attribute identifies a body-format, and the policy attribute contains the policy for that body-format (allowed or disallowed).

4.1.1.5 Extensibility

Other elements from different namespaces MAY be present within a protocol element for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored.

4.1.1.6 Example of a Protocol Element

```
<protocols>
  <protocol name="SIP">
    <methods default-policy="allowed">
       <method name="MESSAGE" policy="disallowed"/>
    </methods>
    <option-tags default-policy="disallowed">
       <option-tag name="100rel" policy="mandatory"/>
       <option-tag name="preconditions" policy="allowed"/>
    </option-tags>
    <feature-tags default-policy="disallowed">
       <feature-tag name="video" policy="allowed"/>
    </feature-tags>
    <bodies default-policy="allowed" default-encryption="allowed">
       <body-disposition name="session" policy="allowed"
                         encryption="disallowed" default-policy="disallowed">
          <body-format name="application/sdp" policy="allowed"/>
       </body-disposition>
    </bodies>
  </protocol>
</protocols>
```

4.1.2 Media Element

The media element contains the policy related to the characteristics of media streams of different types. It has three attributes: maxbandwidth, maxnostreams, and default-policy. They contain the maximum bandwidth the user can count on, the maximum number of media streams that the user is allowed to established at the same time, and the default policy (allowed or disallowed) for stream types that are not listed as stream elements.

The media element contains a series of stream elements.

4.1.2.1 Stream Element

A stream element can have a number of attributes: type, policy, maxbandwidth, and maxnostreams. The type attribute identifies a media type, and the policy attribute contains the policy for that media

type (allowed or disallowed).

The stream element has a number of optional sub-element: the codecs element, the transports element and the directions element.

4.1.2.1.1 Codecs Element

The codecs element contains a default-policy attribute and codec elements. The default-policy attribute contains the policy for codecs that are not listed as codec elements. A codec element can have two attributes: name and policy. The name attribute identifies a codec name, and the policy attribute contains the policy for that codec (allowed, or disallowed). The codec name is the encoding name as defined by the respective RTP profile.

4.1.2.1.2 Transports Element

The transports element contains a default-policy attribute and transport elements. The default-policy attribute contains the policy for transports that are not listed as transport elements. A transport element can have two attributes: name and policy. The name attribute identifies a transport, and the policy attribute contains the policy for that transport (allowed, or disallowed).

4.1.2.1.3 Directions Element

The directions element contains a default-policy attribute and direction elements. The default-policy attribute contains the policy for directions that are not listed as direction elements. A direction element can have two attributes: name and policy. The name attribute identifies a direction (sendrecv, sendonly, recvonly), and the policy attribute contains the policy for that direction (allowed, or disallowed).

4.1.2.1.4 Extensibility

Other elements from different namespaces MAY be present within a stream element for the purposes of extensibility; elements or attributes from unknown namespaces MUST be ignored.

4.1.2.2 Example of a Media Element

```
   <media maxnostreams="4" default-policy="disallowed">
      <stream type="audio" policy="allowed">
           <codecs default-policy="allowed">
               <codec name="PCMU" policy="disallowed"/>
               <codec name="PCMA" policy="disallowed"/>
           </codecs>
           <transports default-policy="disallowed">
               <transport name="RTP/AVP" policy="allowed"/>
           </transports>
           <directions default-policy="disallowed">
               <direction name="sendonly" policy="allowed"/>
           </directions>
       </stream>
   </media>
```

4.2 Schema

The following is the schema for the application/session-policy+xml type:

```
<?xml version="1.0" encoding="UTF-8"?>
TBD
```

4.3 Example

The following is is an example of an application/session-policy+xml document:

```
<?xml version="1.0" encoding="UTF-8"?>
<sessionpolicy xmlns="urn:ietf:params:xml:ns:sessionpolicy"
               version="0"
               domain="example.com"
               entity="sip:alice@example.com">
 <protocols>
  <protocol name="SIP">
    <methods default-policy="allowed"/>
    <option-tags default-policy="allowed"/>
    <feature-tags default-policy="allowed"/>
    <bodies default-policy="allowed" default-encryption="allowed"/>
  </protocol>
 </protocols>
 <media default-policy="allowed"/>
```



```
   </sessionpolicy>
```

5. Security Considerations

Session policy information can be sensitive information. The protocol used to distribute it SHOULD ensure privacy, message integrity and authentication. Furthermore, the protocol SHOULD provide access controls which restrict who can see who else's session policy information.

6. IANA Considerations

This document registers a new MIME type, application/ session-policy+xml, and registers a new XML namespace.

6.1 MIME Registration for application/session-policy+xml

MIME media type name: application
MIME subtype name: session-policy+xml
Mandatory parameters: none
Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [7].
Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [7].
Security considerations: See Section 10 of RFC 3023 [7] and Section 5 of this specification.
Interoperability considerations: none.
Published specification: This document.
Applications which use this media type: This document type has been used to download the session policy of a domain to SIP user agents.
Additional Information:
Magic Number: None
File Extension: .wif or .xml
Macintosh file type code: "TEXT"

Personal and email address for further information: Gonzalo Camarillo, <Gonzalo.Camarillo@ericsson.com>
Intended usage: COMMON
Author/Change controller: The IETF.

6.2 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:sessionpolicy

This section registers a new XML namespace, as per the guidelines in [4]
URI: The URI for this namespace is urn:ietf:params:xml:ns:sessionpolicy.
Registrant Contact: IETF, SIPPING working group,<sipping@ietf.org>, Gonzalo Camarillo, <Gonzalo.Camarillo@ericsson.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>Session Policy Namespace</title>
</head>
<body>
  <h1>Namespace for Session Policy Information</h1>
  <h2>application/session-policy+xml</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

7 References

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

[2] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[3] Hilt, V. and J. Rosenberg, "A Framework for Session-Specific Intermediary Session Policies in SIP", September 2003.

[4] Mealling, M., "The IETF XML Registry", draft-mealling-iana-xmlns-registry-05 (work in progress), June 2003.

[5] Moats, R., "URN Syntax", RFC 2141, May 1997.

[6] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[7] Murata, M., St. Laurent, S. and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[8] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-03 (work in progress), May 2004.

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

[10] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-04 (work in progress), February 2004.

[11] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

Authors' Addresses

Volker Hilt
Bell Labs/Lucent Technologies
101 Crawfords Corner Rd
Holmdel, NJ 07733
USA
EMail: volkerh@bell-labs.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland
EMail: Gonzalo.Camarillo@ericsson.com
Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
East Hanover, NJ 07936
USA
EMail: jdrosen@dynamicsoft.com

Appendix A. Acknowledgements

Many thanks to Allison Mankin and Markus Hofmann for their contributions to this draft.

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.

SIPPING Working Group G. Camarillo
Internet-Draft Ericsson
Expires: November 30, 2004 A. Roach
dynamicsoft
June 2004

Message-Contained URI-Lists in the Session Initiation Protocol (SIP)

draft-ietf-sipping-uri-list-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 30, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes how a user agent can provide another user agent with a list of URIs in a SIP message. The way the receiving user agent uses the URIs in the list is method or status code specific.

Table of Contents

1. Introduction

Some services require a SIP UA (User Agent) to provide another UA (e.g., a SIP URI-list service acting as a UA server) with a set of URIs. For example, a UA creating a conference needs to provide the conference server with the participants. The same way, a UA requesting presence information from a set of users needs to provide the resource list server with the URIs of the users that belong to the list.

These lists are typically configured using out-of-band methods. For instance, a UA can use XCAP [8] to create a list of URIs and to associate this list with a SIP URI (e.g., sip:myfriends@example.com). It can, then, send a SIP request (an INVITE or a SUBSCRIBE in our previous examples) to that SIP URI.

Still, there is a need to create lists of URIs and send them directly in a SIP message. Transporting the URI list in the SIP message that triggers the service usually helps reduce the service establishment time, and is useful for UAs that do not have access to a server to host their list (and they cannot act as a server themselves).

In any case, the way the application server interprets the URI list received in the request is method specific.

A UA creating a SIP request or response that needs to carry a URI list places the URI list (e.g., an XCAP resource list [4]) in a body part whose disposition type is "uri-list". The way the receiving UA interprets the URI list received is method specific, or, in the case of a response, status code specific.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

3. The uri-list Disposition Type

We define a new disposition type for the Content-Disposition header field: uri-list. Both requests and responses MAY carry uri-list bodies.

Bodies whose disposition type is uri-list carry a list of URIs. The way a UA receiving a URI list interprets it is method specific, or, in the case of a response, status code specific.

3.1 Default URI List Format

The default format for uri-list bodies is the XCAP resource list format defined in [4]. So, SIP entities handling uri-list bodies MUST support this format.

Nevertheless, the XCAP resource list format provides features such as hierarchical lists and list's attributes that are not needed by many services, which only need to transfer a flat list of URIs between two UAs. The amount of information that a URI list needs to carry between two UAs is method or status code specific. Additionally, the way a client and a server negotiate the amount of information needed for a particular service is method specific as well.

A client invoking a particular service SHOULD NOT include more information in its URI list than the service requires. A server providing a particular service MAY discard any extra information which is received in a URI list from the client.

The following is an example of a flat list without attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com" />
    <entry uri="sip:joe@example.org" />
    <entry uri="sip:ted@example.net" />
  </list>
</resource-lists>
```

Figure 1: URI List

4. Pointing to External URI Lists

UAs that want to use an external URI list, instead of sending it as a body part, SHOULD use the content indirection mechanism defined in [5]. Indirected body parts are equivalent and have the same treatment as in-line body parts.

5. Example

The following is an example of an INVITE request that carries a URI list in its body. The Request-URI of this INVITE contains a pointer to the body part carrying the list.

```
INVITE sip:conf-fact@example.com SIP/2.0
Via: SIP/2.0/TCP client.chicago.example.com
    ;branch=z9hG4bKhjhs8ass83
Max-Forwards: 70
To: Conf Factory <sip:conf-fact@example.com>
From: Carol <sip:carol@chicago.example.com>;tag=32331
Call-ID: d432fa84b4c76e66710
CSeq: 1 INVITE
Contact: <sip:carol@client.chicago.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
     SUBSCRIBE, NOTIFY
Allow-Events: dialog
Accept: application/sdp, message/sipfrag,
Conten-Type: multipart/mixed;boundary="boundary1"
Content-Length: 635

--boundary1
Content-Type: application/sdp

v=0
o=carol 2890844526 2890842807 IN IP4 chicago.example.com
s=Example Subject
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 20002 RTP/AVP 31
a=rtpmap:31 H261/90000

--boundary1
Content-Type: application/resource-lists+xml
Content-Disposition: uri-list

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com" />
    <entry uri="sip:joe@example.org" />
    <entry uri="sip:ted@example.net" />
  </list>
</resource-lists>
--boundary1--
```

Figure 2: INVITE request

Refer to (draft-ietf-sipping-uri-list-conferencing-00.txt) for the normative details on how a list can be used with the INVITE method.

6. Security Considerations

This document discusses how to carry URI lists in SIP messages. Attackers may attempt to modify URI lists sent between two user agents. This would cause a different service behavior than expected by the user agents. To prevent this attack, user agents SHOULD integrity protect URI lists using mechanisms such as S/MIME, which can also provide URI list confidentiality, if needed.

Some application servers, on reception of a SIP message with a URI list, send SIP requests to the URIs in the list. These application servers are referred to as SIP URI-list services. The Security Considerations Section of the Requirements and Framework for SIP SIP URI-List Services [6] discusses issues related to SIP URI-list services. Implementations of SIP URI-list services MUST follow the security-related rules in [6]. These rules include mandatory authentication and authorization of clients, and opt-in lists.

7. IANA Considerations

This document defines a new Content-Disposition header field disposition type (uri-list) in Section 3. This value should be registered in the IANA registry for Content-Dispositions on

http://www.iana.org/assignments/mail-cont-disp

with the following description:

uri-list the body contains a list of URIs

8. Acknowledges

Alan Johnston, Orit Levin, and Cullen Jennings provided useful comments on this document.

9. References

9.1 Normative References

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

[2] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, August 1998.

[3] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[4] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[5] Olson, S., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", draft-ietf-sip-content-indirect-mech-03 (work in progress), June 2003.

[6] Camarillo, G., "Requirements for Session Initiation Protocol (SIP) Exploder Invocation", draft-camarillo-sipping-exploders-02 (work in progress), February 2004.

9.2 Informational References

[7] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[8] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-02 (work in progress), February 2004.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Adam Roach
dynamicsoft
5100 Tennyson Pkwy
Suite 1200
Plano, TX 75024
US

EMail: adam@dynamicsoft.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.

SIPPING Working Group
Internet-Draft
Expires: January 5, 2005

G. Camarillo
Ericsson
A. Johnston
MCI
July 7, 2004

Conference Establishment Using Request-Contained Lists in the Session Initiation Protocol (SIP)

draft-ietf-sipping-uri-list-conferencing-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 5, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes how to create a conference using SIP URI-list services. In particular, we describe a mechanism that allows a client to provide a conference server with the initial list of participants using an INVITE-contained URI-list.

Table of Contents

1. Introduction

Section 4.5 of [3] describes how to create a conference using ad-hoc SIP [2] methods. The client sends an INVITE request to a conference factory URI, and receives the actual conference URI, which contains the "IsFocus" feature tag, in the Contact header field of a response (typically a 200 OK).

Once the client obtains the conference URI, it can add participants to the newly created conference in several ways, which are described in [3].

Some environments have tough requirements regarding conference establishment time. So, they require the client to be able to request the creation of an ad-hoc conference and to provide the server with the initial set of participants in a single operation. This document describes how to meet this requirement using the mechanism to transport URI lists in SIP messages described in [4].

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

3. Providing a Conference Server with a URI-List

A client that wants to include the set of initial participants in its initial INVITE to create an ad-hoc conference, adds a body whose disposition type is uri-list, as defined in [4], with a URI list that contains the participants that the client wants the server to INVITE. The client sends this INVITE to the conference factory URI.

4. URI List Format

As described in [4], the default format for URI lists in SIP is the XCAP resource list format [5]. Still, specific services need to describe which information clients should include in their URI lists, as described in [4].

Conferencing UAs SHOULD use flat lists (i.e., no hierarchical lists), SHOULD NOT use any entry's attributes but "uri", and SHOULD NOT include any elements inside entries but "display-name" elements.

A conference factory application receiving a URI list with more information than what we have just described SHOULD discard all the

extra information.

5. Conference Server Behavior

On reception of an INVITE with a uri-list body as described in Section 3, a conference server MUST follow the rules described in [3] to create ad-hoc conferences. Once the ad-hoc conference is created, the conference server SHOULD attempt to add the participants in the URI list to the conference as if their addition had been requested using any of the methods described in [3] (e.g., using CPCP [7]).

Once the conference server has created the ad-hoc conference and has attempted to add the initial set of participants, the conference server behaves as a regular conference server and MUST follow the rules in [3].

Note that the status code in the response to the INVITE does not provide any information about whether or not the conference server was able to bring the users in the URI-list into the conference. That is, a 200 (OK) means that the conference was created successfully, that the client that generated the INVITE is in the conference, and that the server understood the URI list. If the client wishes to obtain information about the status of other users in the conference it SHOULD use general conference mechanisms, such as the conference package [8].

6. Example

The following is an example of an INVITE request, which carries a URI list in a uri-list body, sent by a UA to a conference factory application.

```
INVITE sip:conf-fact@example.com SIP/2.0
Via: SIP/2.0/TCP client.chicago.example.com
    ;branch=z9hG4bKhjhs8ass83
Max-Forwards: 70
To: Conf Factory <sip:conf-fact@example.com>
From: Carol <sip:carol@chicago.example.com>;tag=32331
Call-ID: d432fa84b4c76e66710
CSeq: 1 INVITE
Contact: <sip:carol@client.chicago.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
     SUBSCRIBE, NOTIFY
Allow-Events: dialog
Accept: application/sdp, message/sipfrag,
Conten-Type: multipart/mixed;boundary="boundary1"
Content-Length: 635
```

```
--boundary1
Content-Type: application/sdp

v=0
o=carol 2890844526 2890842807 IN IP4 chicago.example.com
s=Example Subject
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 20002 RTP/AVP 31
a=rtpmap:31 H261/90000

--boundary1
Content-Type: application/resource-lists+xml
Content-Disposition: uri-list

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com" />
    <entry uri="sip:joe@example.org" />
    <entry uri="sip:ted@example.net" />
  </list>
</resource-lists>
--boundary1--
```

Figure 1: INVITE request

7. Security Considerations

This document discusses setup of SIP conferences using a request-contained URI-list. Both conferencing and URI-lists services have specific security requirements which will be summarized here. Conferences generally have authorization rules about who may or may not join a conference, what type of media may or may not be used, etc. This information is used by the focus to admit or deny participation in a conference. It is RECOMMENDED that these types of authorization rules be used to provide security for a SIP conference.

For this authorization information to be used, the focus needs to be able to authenticate potential participants. Normal SIP mechanisms including Digest authentication and certificates can be used. These conference specific security requirements are discussed further in the requirements and framework documents.

For conference creation using a list, there are some additional security considerations. The Security Considerations Section of the Requirements and Framework for SIP URI-List Services [6] discusses issues related to SIP URI-list services. Given that a conference server sending INVITEs to a set of users acts as an URI-list service, implementations of conference servers that handle lists MUST follow the security-related rules in [6]. These rules include mandatory authentication and authorization of clients, and opt-in lists.

8. Acknowledges

Cullen Jennings provided useful comments on this document.

9. References

9.1 Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Johnston, A. and O. Levin, "Session Initiation Protocol Call Control - Conferencing for User Agents", draft-ietf-sipping-cc-conferencing-03 (work in progress), February 2004.

[4] Camarillo, G., "Providing a Session Initiation Protocol (SIP) Application Server with a List of URIs", draft-camarillo-sipping-uri-list-01 (work in progress), February 2004.

[5] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[6] Camarillo, G., "Requirements for Session Initiation Protocol (SIP) Exploder Invocation", draft-camarillo-sipping-exploders-02 (work in progress), February 2004.

9.2 Informational References

[7] Koskelainen, P. and H. Khartabil, "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Conference Policy Manipulation", draft-koskelainen-xcon-xcap-cpcp-usage-02 (work in progress), February 2004.

[8] Rosenberg, J. and H. Schulzrinne, "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-03 (work in progress), February 2004.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Alan Johnston
MCI
100 South 4th Street
St. Louis, MO 63102
USA

EMail: alan.johnston@mci.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.

SIPPING Working Group M. Garcia-Martin
Internet-Draft Nokia
Expires: January 5, 2005 G. Camarillo
Ericsson
July 7, 2004

Multiple-Recipient MESSAGE Requests in the Session Initiation Protocol (SIP)

draft-ietf-sipping-uri-list-message-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 5, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document specifies how to request a SIP URI-list service to send a copy of a MESSAGE to a set of destinations. The client sends a SIP MESSAGE request with a URI-list to the URI-list service, which sends a similar MESSAGE request to each of the URIs included in the list.

Table of Contents

1. Introduction

SIP [2] can carry instant messages in MESSAGE [3] requests. The Advanced Instant Messaging Requirements for SIP [8] mentions the need for sending a MESSAGE request to multiple receipients:

"REQ-GROUP-3: It MUST be possible for a user to send to an ad-hoc group, where the identities of the recipients are carried in the message itself."

To meet this requirement, we allow SIP MESSAGE requests carry URI-lists in "uri-list" body parts, as specified in [4]. A SIP URI-list service, which is a specialized application server, receives the request and sends a similar MESSAGE request to each of the URIs in the list. Each of these MESSAGE requests contains a copy of the body included in the original MESSAGE request.

The UAC (User Agent Client) needs to be configured with the SIP URI of the application server that provides the functionality. Discovering and provisioning of this URI to the UAC is outside the scope of this document.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

'MESSAGE URI-list service': SIP application server that receives a MESSAGE request with a URI-list and sends a similar MESSAGE request to each URI in the list. MESSAGE URI-list services behave effectively as specialised B2BUAs (Back-To-Back-User-Agents). A MESSAGE URI-list service can also offer URI-list services for other methods, although this functionality is outside the scope of this document. In this document we only discuss MESSAGE URI-list services.

'Incoming MESSAGE request': A SIP MESSAGE request that a UAC creates and addresses to a MESSAGE URI-list service. Besides the regular instant message payload, an incoming MESSAGE request contains a URI-list.

'Outgoing MESSAGE request': A SIP MESSAGE request that a MESSAGE URI-list service creates and addresses to a UAS (User Agent Server). It contains the regular instant message payload.

3. Procedures at the UAC

A client that wants to create a multiple-recipient MESSAGE request adds a body part, whose disposition type is "uri-list", which contains a URI-list with the recipients of the MESSAGE.

Multiple-recipient MESSAGE requests typically contain a multipart body that contains the body carrying the list and the actual instant message payload. In some cases, the MESSAGE request may contain bodies other than the text and the list bodies (e.g., when the request is protected with S/MIME [6]).

Typically, the MESSAGE URI-list service will copy all the significant header fields in the outgoing MESSAGE request. However, there might be cases where the SIP UA wants the MESSAGE URI-list service to add a particular header field with a particular value, even if the header field wasn't present in the MESSAGE request sent by the UAC. In this case, the UAC MAY use the "?" mechanism described in Section 19.1.1 of RFC 3261 [2] to encode extra information in any URI in the list. However, the UAC MUST NOT use the special "body" hname (see Section 19.1.1 of RFC 3261 [2]) to encode a body, since the body is present in the MESSAGE request itself.

The following is an example of a URI that uses the "?" mechanism:

sip:bob@example.com?Accept-Contact=*%3bmobility%3d%22mobile%22

The previous URI requests the MESSAGE URI-list service to add the following header field to a MESSAGE request to be sent to bob@example.com:

Accept-Contact: *;mobility="mobile"

As described in [4], the default format for URI-lists in SIP is the XCAP resource list format [5]. Still, specific services need to describe which information clients should include in their URI lists, as described in [4]

UAs generating multiple recipient MESSAGEs SHOULD use flat lists (i.e., no hierarchical lists), SHOULD NOT use any entry's attributes but "uri", and SHOULD NOT include any elements inside entries but "display-name" elements.

A MESSAGE URI-list service receiving a URI-list with more information than what we have just described SHOULD discard all the extra information.

4. Procedures at the MESSAGE URI-List Service

On reception of a MESSAGE request with a URI-list, a MESSAGE URI-list service SHOULD answer to the UAC with a 202 Accepted response. Note that the status code in the response to the MESSAGE does not provide any information about whether or not the MESSAGEs generated by the URI-list service were successfully delivered to the URIs in the list. That is, a 202 Accepted means that the MESSAGE URI-list service has received the MESSAGE and that it will try to send a similar MESSAGE to the URIs in the list. Designing a mechanism to inform a client about the delivery status of an instant message is outside the scope of this document.

On reception of a MESSAGE request with a URI-list, a MESSAGE URI-list service SHOULD create as many new MESSAGE requests as URIs the list contains, except when two of those URIs are equivalent (section 19.1.4 of RFC 3261 [2] defines equivalent URIs), in which case the MESSAGE URI-list service SHOULD create only one outgoing MESSAGE request per URI.

When creating the body of each of the outgoing MESSAGE requests, the MESSAGE URI-list service tries to keep the relevant bodies of the incoming MESSAGE request and copies them to the outgoing MESSAGE request. The following guidelines are provided:

- The incoming MESSAGE request typically contains a URI-list body [4] with the actual list of recipients. The MESSAGE URI-list service need not copy the URI-list body to each of the outgoing MESSAGE requests, although it MAY do it.
  NOTE: This document does not provide any semantics associated to a URI-list body included in an outgoing MESSAGE request. Future extensions may indicate actions at a UAS when it receives that body.
- A MESSAGE request received at a MESSAGE URI-list service can contain one or more security bodies encrypted with the public key of the MESSAGE URI-list service. These bodies are deemed to be read by the URI-list service rather than the recipient of the outgoing MESSAGE request (which will not be able to decrypt them). Therefore, a MESSAGE URI-list service MUST NOT copy any security body (such as an S/MIME encrypted body) addressed to the MESSAGE URI-list service to the outgoing MESSAGE request. This includes bodies encrypted with the public key of the URI-list service.
- An exception to this rule is the URI-list itself: as mentioned in Section 4, a MESSAGE URI-list service need not, but MAY, copy the URI-list into each of the outgoing MESSAGE requests; on doing so, a MESSAGE URI-list service SHOULD use S/MIME [6] to encrypt the URI-list with the public key of the receiver.

- The MESSAGE URI-list service SHOULD copy all the rest of the message bodies (e.g., text messages, images, etc.) to the outgoing MESSAGE request.
- If there is only one body left, the MESSAGE URI-list service MUST remove the multipart/mixed wrapper in the outgoing MESSAGE request.

The rest of the MESSAGE request corresponding to a given URI in the list MUST be created following the rules in Section 19.1.5 "Forming Requests from a URI" of RFC 3261 [2]. In particular, Section 19.1.5 of RFC 3261 [2] states:

"An implementation SHOULD treat the presence of any headers or body parts in the URI as a desire to include them in the message, and choose to honor the request on a per-component basis."

SIP allows to append a "method" parameter to a URI. Therefore, it is legitimate that an the "uri" attribute of the "entry" element in the XCAP resource list contains a "method" parameter. MESSAGE URI-list services MUST generate only MESSAGE requests, regardless of the "method" parameter that the URIs in the list indicate. Effectively, MESSAGE URI-list services MUST ignore the "method" parameter in each of the URIs present in the URI list.

It is RECOMMENDED that the MESSAGE URI-list service copies the From header field of the incoming MESSAGE into the outgoing MESSAGE requests (note that this does not apply to the "tag" parameter). The MESSAGE URI-list service SHOULD also copy into the outgoing MESSAGE request any P-Asserted-Identity header fields present in the incoming MESSAGE request.

For each given outgoing MESSAGE request, the MESSAGE URI-list service SHOULD generate a new To header field value which, according to the procedures of RFC 3261 Section 8.1.1.1, should be equal to the Request-URI of the outgoing MESSAGE request.

For each given outgoing MESSAGE request, the MESSAGE URI-list service SHOULD initialize the values of the Call-ID, CSeq and Max-Forwards header fields. The MESSAGE URI-list service should also include its own value in the Via header field.

5. Examples

The following is an example of an incoming MESSAGE request which carries a URI list in its body.

```
MESSAGE sip:list-service.example.com SIP/2.0
Via: SIP/2.0/TCP uac.example.com
    ;branch=z9hG4bKhjhs8ass83
Max-Forwards: 70
To: MESSAGE URI-List Service <sip:list-service.example.com>
From: Carol <sip:carol@example.com>;tag=32331
Call-ID: d432fa84b4c76e66710
CSeq: 1 MESSAGE
Content-Type: multipart/mixed;boundary="boundary1"
Content-Length: 440

--boundary1
Content-Type: text/plain

Hello World!

--boundary1
Content-Type: application/resource-lists+xml
Content-Disposition: uri-list

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com" />
    <entry uri="sip:joe@example.org" />
    <entry uri="sip:ted@example.net" />
  </list>
</resource-lists>
--boundary1--
```

Figure 3: Multiple recipient incoming MESSAGE request

The following is an example of one of the outgoing MESSAGE requests that the MESSAGE URI-list service creates.


```
MESSAGE sip:bill@example.com SIP/2.0
Via: SIP/2.0/TCP list-service.example.com
    ;branch=z9hG4bKhjhs8as34sc
Max-Forwards: 70
To: <sip:bill@example.com>
From: Carol <sip:carol@uac.example.com>;tag=210342
Call-ID: 39s02sdsl20d9sj2l
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 13

Hello World!
```

Figure 4: Outgoing MESSAGE request

6. Security Considerations

The Security Considerations Section of the Requirements and Framework for SIP URI-List Services [7] discusses issues related to SIP URI-list services. Implementations of MESSAGE URI-list services MUST follow the security-related rules in [7]. These rules include mandatory authentication and authorization of clients, and opt-in lists.

If the contents of the instant message needs to be kept private, the user agent client SHOULD use S/MIME [6] to prevent a third party from viewing this information. In this case, the user agent client SHOULD encrypt the instant message body with a content encryption key. Then, for each receiver in the list, the UAC SHOULD encrypt the content encryption key with the public key of the receiver, and attach it to the MESSAGE request.

7. Acknowledgements

Duncan Mills supported the idea of having 1 to n MESSAGEs. Ben Campbell, Paul Kyzivat, and Cullen Jennings provided helpful comments.

8. Change control

8.1 Changes from draft--sipping-message-exploder-00.txt to draft-ietf-sipping-uri-list-message-00.txt

Clarified that the MESSAGE exploder should not distribute a body that has been encrypted with the public key of the exploder. The exception is the URI list, which can be distributed by the exploder, providing that is encrypted with the public key of the receiver.

The security considerations section describes how to encrypt the list and how to encrypt the instant message payload.

Terminology aligned with the requirements and the framework for URI-list services (e.g., the term "exploder" has been deprecated).

8.2 Changes from draft-garcia-simple-message-exploder-00.txt to draft-garcia-sipping-message-exploder-00.txt

The MESSAGE exploder may or may not copy the URI list body to the outgoing MESSAGE request. This allows to extend the mechanism with a Reply-to-all feature.

It is clarified that the MESSAGE exploder must not include a list in the outgoing MESSAGE requests. This avoids loops or requires a MESSAGE exploder functionality in the next hop.

The MESSAGE exploder must remove the multipart/mixed wrapper if there is only one body left in the outgoing MESSAGE request.

Filename changed due to focus on the SIPPING WG.

9. References

9.1 Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[4] Camarillo, G., "Providing a Session Initiation Protocol (SIP) Application Server with a List of URIs", draft-camarillo-sipping-uri-list-01 (work in progress), February 2004.

[5] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[6] Ramsdell, B., "S/MIME Version 3.1 Message Specification", draft-ietf-smime-rfc2633bis-07 (work in progress), February 2004.

[7] Camarillo, G., "Requirements for Session Initiation Protocol (SIP) Exploder Invocation", draft-camarillo-sipping-exploders-02 (work in progress), February 2004.

9.2 Informational References

[8] Rosenberg, J., "Advanced Instant Messaging Requirements for the Session Initiation Protocol (SIP)", draft-rosenberg-simple-messaging-requirements-01 (work in progress), February 2004.

[9] Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-02 (work in progress), July 2003.

[10] Jennings, C., Peterson, J. and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002.

Authors' Addresses

Miguel A. Garcia-Martin
Nokia
P.O.Box 407
NOKIA GROUP, FIN 00045
Finland

EMail: miguel.an.garcia@nokia.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.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.

SIPPING Working Group G. Camarillo
Internet-Draft Ericsson
Expires: November 30, 2004 A. Roach
dynamicsoft
June 2004

Subscriptions to Request-Contained Resource Lists in the Session Initiation Protocol (SIP)

draft-ietf-sipping-uri-list-subscribe-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 30, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes how to create subscriptions to request-contained resource lists in SIP. This is done by having the SUBSCRIBE request that requests the creation of the subscription carry a URI list.

Table of Contents

1. Introduction

Subscriptions to homogeneous resource lists in SIP [2] are described in [3], which assumes that a resource list (i.e., a list of URIs) is represented by a URI (generally a SIP URI). Once a UA obtains the URI that represents a resource list, it can use the mechanisms described in [3] to subscribe to it.

For example, let us assume that the resource list identified by the SIP URI sip:my-friends@example.com contains the following URIs:

```
sip:bill@example.com
sip:joe@example.org
sip:ted@example.net
```

If a UA subscribes to the presece information of sip:my-friends@example.com, it will obtain the presence information of all the resources in the list.

List creation is outside the scope of [3]. This document describes a way to create a list with a set of resources, and subscribe to it, using a single SIP request. We use the mechanism to carry URI lists in SIP messages described in [4].

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

3. Providing a Resource List Server with a URI List

A client that wants to create a resource list and subscribe to it using the mechanism described in this document generates a SUBSCRIBE with a body whose disposition type is uri-list as defined in [4]. This body contains the URIs that belong to the resource list. The client MUST build the remaining of the SUBSCRIBE request following the rules in [3].

4. URI List Format

As described in [4], the default format for URI lists in SIP is the XCAP resource list format [6]. Still, specific services need to describe which information clients should include in their URI lists, as described in [4].

UAs subscribing to a request-contained resource list SHOULD use flat lists (i.e., no hierarchical lists), SHOULD NOT use any entry's attributes but "uri", and SHOULD NOT include any elements inside entries but "display-name" elements.

Resource list servers receiving a URI list with more information than what we have just described SHOULD discard all the extra information.

5. Resource List Server Behavior

On reception of a SUBSCRIBE with a URI list as described in Section 3, a resource list server MUST follow the rules described in [3] to create the subscription, using the URI list just received as the resource list for the subscription.

Once the resource list server has created the subscription, it behaves as a regular resource list server and MUST follow the rules in [3].

Note that the status code in the response to the SUBSCRIBE does not provide any information about whether or not the resource list server was able to succesfully subscribe to the URIs in the URI list. The client obtains this information in the NOTIFIES sent by the server.

6. Resource List Life-Time

The life-time of a resource list created as described in Section 5 is blundled to the life-time of the subscription. That is, the resource list SHOULD be destroyed when the subscription expires or is otherwise terminated.

7. Providing a URI to Manipulate a Resource List

A client may need to manipulate a resource list at a resource list server. The resource list server MAY provide a URI to manipulate the resource list associated with a subscription using the Call-Info header field in the NOTIFY that establishes the subscription. The "purpose" parameter of the Call-Info header field MUST have a value of "list-management", which we register with the IANA in Section 10. The following is an example of such a header field.

```
Call-Info: <http://xcap.example.com/your-list.xml>
           ;purpose=list-management
```

8. Example

The following is an example of a SUBSCRIBE request, which carries a URI list in its body, sent by a UA to a resource list server.

```
SUBSCRIBE  sip:rls@example.com SIP/2.0
Via: SIP/2.0/TCP terminal.example.com;branch=z9hG4bKwYb6QREiCL
Max-Forwards: 70
To: RLS <sip:rls@example.com>
From: <sip:adam@example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.example.com
CSeq: 1 SUBSCRIBE
Contact: <sip:terminal.example.com>
Event: presence
Expires: 7200
Supported: eventlist
Accept: application/cpim-pidf+xml
Accept: application/rlmi+xml
Accept: multipart/related
Accept: multipart/signed
Accept: multipart/encrypted
Content-Type: application/resource-lists+xml
Content-Disposition: uri-list
Content-Length: 274

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com" />
    <entry uri="sip:joe@example.org" />
    <entry uri="sip:ted@example.net" />
  </list>
</resource-lists>
```

Figure 1: SUBSCRIBE request

9. Security Considerations

The Security Considerations Section of [3] discusses security issues related to resource list servers. Resource list servers accepting request-contained URI-lists MUST also follow the security guidelines given in [3].

The Security Considerations Section of the Requirements and Framework for SIP URI-List Services [5] discusses issues related to SIP URI-list services. Given that a resource list server sending

SUBSCRIBEs to a set of users acts as a URI-list service, implementations of resource list servers that handle request-contained URI-lists MUST follow the security-related rules in [5]. These rules include mandatory authentication and authorization of clients, and opt-in lists.

10. IANA Considerations

The document defines the "list-management" value for the purpose parameter of the Call-Info header field. A reference to this RFC should be added to the purpose Call-Info parameter entry in the SIP header field parameter registry on:

http://www.iana.org/assignments/sip-parameters

11. Acknowledges

Cullen Jennings provided useful comments on this document.

12. References

12.1 Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Roach, A., Rosenberg, J. and B. Campbell, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", draft-ietf-simple-event-list-04 (work in progress), June 2003.

[4] Camarillo, G., "Providing a Session Initiation Protocol (SIP) Application Server with a List of URIs", draft-camarillo-sipping-uri-list-01 (work in progress), February 2004.

[5] Camarillo, G., "Requirements for Session Initiation Protocol (SIP) Exploder Invocation", draft-camarillo-sipping-exploders-02 (work in progress), February 2004.

12.2 Informational References

[6] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists",

draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Adam Roach
dynamicsoft
5100 Tennyson Pkwy
Suite 1200
Plano, TX 75024
US

EMail: adam@dynamicsoft.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.

SIPPING Working Group G. Camarillo
Internet-Draft Ericsson
Expires: January 5, 2005 July 7, 2004

Requirements and Framework for Session Initiation Protocol (SIP) Uniform Resource Identifier (URI)-List Services

draft-ietf-sipping-uri-services-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 5, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes the need for SIP URI-List Services and provides requirements for their invocation. Additionaly, it defines a framework which includes all the SIP extensions needed to meet these requirements.

Table of Contents

1. Introduction

Some applications require that, at a given moment, a SIP [2] UA (User Agent) performs a similar transaction with a number of remote UAs. For example, an instant messaging application that needs to send a particular message (e.g., "Hello folks") to n receivers needs to send n MESSAGE requests; one to each receiver.

When the transacton that needs to be repeated consists of a large request, or the number of recipients is high, or both, the access network of the UA needs to carry a considerable amount of traffic. Completing all the transactions on a low-bandwidth access would require a long time. This is unacceptable for a number of applications.

A solution to this problem consists of introducing URI-list services in the network. The task of a SIP URI-list service is to receive a request that contains or references a URI-list and send a number of similar requests to the destinations in this list. Once the requests are sent, the URI-list service typically informs the UA about their status. Effectively, the URI-list service behaves as a B2BUA (Back-To-Back-User-Agent).

If the request references an external URI-list (e.g., the Request-URI is a SIP URI which is associated with a URI-list at the server), this URI-list is referred to as an stored URI-list. If the request contains the URI-list, the URI-list is referred to as a request-contained URI-list.

Stored URI-lists are typically set up using out-of-band mechanisms (e.g., XCAP [9]). An example of a URI-list service for SUBSCRIBE requests that uses stored URI-lists is described in [4].

The Advanced Instant Messaging Requirements for SIP [5] mentions the need for request-contained URI-list services for MESSAGE transactions

"REQ-GROUP-3: It MUST be possible for a user to send to an ad-hoc group, where the identities of the recipients are carried in the message itself."

The remainder of this document provides requirements for both stored and request-contained SIP URI-list services.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as

described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

3. Requirements

Section 3.1 discusses requirements that only apply to request-contained URI-list services and Section 3.2 discusses requirements that apply to both stored and request-contained URI-list services.

3.1 Requirements for Request-Contained URI-List Services

1. The URI-list service invocation mechanism MUST allow the invoker to provide a list of destination URIs to the URI-list service. This URI-list MAY consist of one or more URIs.
2. The mechanism to provide the URI-list to the URI-list service MUST NOT be request specific.
3. The invocation mechanism SHOULD NOT require more than one RTT (Round-Trip Time).

3.2 General Requirements for URI-List Services

1. An URI-list service MAY include services beyond sending requests to the URIs in the URI-list. That is, URI-list services can be modelled as application servers. For example, a URI-list service handling INVITE requests may behave as a conference server and perform media mixing for all the participants.
2. The interpretation of the meaning of the URI-list sent by the invoker MUST be at the discretion of the application to which the list is sent.
3. It MUST be possible for the invoker to find out about the result of the operations performed by the URI-list service with the URI-list. An invoker may, for instance, be interested in the status of the transactions initiated by the URI-list service.
4. URI-list services MUST NOT send requests to multiple destinations without authenticating the invoker.

4. Framework

Although Section 3 contains specific requirements for SIP URI-list services, this framework is not restricted to application servers that only provide request fan-out services. Per the general requirement number 1, we also deal with application servers that provide a particular service that includes a request fan-out (e.g., a conference server that INVITEs several participants which are chosen by a user agent).

4.1 Carrying URI-Lists in SIP

The requirements that relate to request-contained URI-list services identify the need for a request-independent mechanism to provide a SIP URI-list service with a URI-list in a single RTT. The mechanism described in (draft-ietf-sipping-uri-list-00.txt) [3] meets these three requirements.

UAs (User Agents) use body parts whose disposition type is uri-list to transport URI-lists. The default URI-list format for SIP entities is the XCAP resource list format defined in [6].

4.2 Processing of URI-Lists

According to the general requirements 1 and 2, URI-list services can behave as application servers. That is, taking a URI-list as an input, they can provide arbitrary services.

So, the interpretation of the URI-list by the server depends on the service to be provided. For example, for a conference server, the URIs in the list may identify the initial set of participants. On the other hand, for a server dealing with MESSAGEs, the URIs in the list may identify the recipients of an instant message.

At the SIP level, this implies that the behavior of application servers receiving requests with URI-lists SHOULD be specified on a per method basis. Examples of such specifications are [draft-ietf-sipping-uri-list-conferencing-00.txt] for INVITE, [draft-ietf-sipping-multiple-refer-00.txt] for REFER, [draft-ietf-sipping-uri-list-message-00.txt] for MESSAGE, and [draft-ietf-sipping-uri-list-subscribe-00.txt] for SUBSCRIBE.

4.3 Results

According to requirement 6, user agents should have a way to obtain information about the operations performed by the application server. Since these operations are service specific, the way user agents are kept informed is also service specific. For example, a user agent establishing an adhoc conference with an INVITE with a URI-list may discover which participants were successfully brought in into the conference by using the conference package [8].

5. Security Considerations

Security plays an important role in the implementation of any URI-list service. By definition, a URI-list service takes one request in and sends a potentially large number of them out. Attackers may attempt to use URI-list services as traffic amplifiers to launch DoS

attacks. In addition, malicious users may attempt to use URI-list services to distribute unsolicited messages (i.e., SPAM) or to make unsolicited VoIP calls. This section provides guidelines to avoid these attacks.

5.1 List Integrity and Confidentiality

Attackers may attempt to modify URI-lists sent from clients to servers. This would cause a different behavior at the server than expected by the client (e.g., requests being sent to different recipients as the ones specified by the client). To prevent this attack, clients SHOULD integrity protect URI-lists using mechanisms such as S/MIME, which can also provide URI-list confidentiality if needed.

5.2 Amplification Attacks

URI-list services take a request in and send a potentially large number of them out. Given that URI-list services are typically implemented on top of powerful servers with high-bandwidth access links, we should be careful to keep attackers from using them as amplification tools to launch DoS (Denial of Service) attacks.

Attackers may attempt to send a URI-list containing URIs whose host parts route to the victims of the DoS attack. These victims do not need to be SIP nodes; they can be non-SIP endpoints or even routers. If this attack is successful, the result is that an attacker can flood with traffic a set of nodes, or a single node, without needing to generate a high volume of traffic itself.

> Note, in any case, that this problem is not specific to SIP URI-list services; it also appears in scenarios which relate to multihoming where a server needs to contact a set of IP addresses provided by a client (e.g., an SCTP [10] endpoint using HEARTBEATs to check the status of the IP addresses provided by its peer at association establishment).

There are several measures that need to be taken to prevent this type of attack. The first one is keeping unauthorized users from using URI-list services. So, URI-list services MUST NOT perform any request explosion for an unauthorized user. URI-list services MUST authenticate users and check whether they are authorized to request the service before performing any request fan-out.

> Note that the risk of this attack also exists when a client uses stored URI-lists. Application servers MUST use authentication and authorization mechanisms with equivalent security properties when dealing with stored and request-contained URI-lists.

Even though the previous rule keeps unauthorized users from using URI-list services, authorized users may still launch attacks using a these services. To prevent these attacks, we introduce the concept of opt-in lists. That is, URI-list services should not allow a client to place a user (identified by his or her URI) in a URI-list unless the user has previously agreed to be placed in such a URI-list. So, URI-list services MUST NOT send a request to a destination which has not agreed to receive requests from the URI-list service beforehand. Users can agree to receive requests from a URI-list service in several ways, such as filling a web page, sending an email, or signing a contract. Additionally, users MUST be able to further describe the requests they are willing to receive. For example, a user may only want to receive requests from a particular URI-list service on behalf of a particular user. Effectively, these rules make URI-lists used by URI-list services opt-in lists.

When a URI-list service receives a request with a URI-list from a client, the URI-list service checks whether all the destinations have agreed beforehand to receive requests from the service on behalf of this client. If the URI-list has permission to send requests to all of the targets in the request, it does so. If not, the URI-list service rejects the request, indicating in the rejection the set of targets for which it did not have permission. This allows the client to request permission for those targets.

DoS amplification would still happen if the URI-list service automatically contacted the full set of targets for which it did not have permission in order to request permission. The URI-list service would be receiving one SIP request and sending out a number of authorization request messages. In order to avoid this amplification, the URI-list service must ensure that the client generates roughly the same amount of traffic towards the URI-list service as the service generates towards the destinations. Consequently, the URI-list service MUST require that clients send and individual authorization request for each destination.

These individual authorization requests sent by the client may or may not be routed through the URI-list service. In any case, the URI-list service MUST be informed about the destinations' responses to these authorization requests in order to authorize requests towards them. One possible mechanism for clients to send authorization requests to the destinations is specified in [draft-rosenberg-sipping-consent-framework-00.txt], which discusses consent-based communications in SIP. The requirements for consent-based communications in SIP are discussed in [draft-rosenberg-sipping-consent-reqs-00.txt]

5.3 Unsolicited Requests

Opt-in lists should help fighting SPAMMERS. Still, if a URI-list service is used to send unsolicited requests to one or several destinations, it should be possible to track down the sender of such requests. To do that, URI-list services MAY provide information about the identity of the original sender of the request in their outgoing requests. URI-list services can use Authenticated Identity Bodies (AIB) [7] to provide this information.

5.4 General Issues

URI-list services MAY have policies that limit the number of URIs in the lists they accept, as a very long list could be used in a denial of service attack to place a large burden on the URI-list service to send a large number of SIP requests.

The general requirement number 4, which states that URI-list services need to authenticate their clients, and the previous rules apply to URI-list services in general. In addition, specifications dealing with individual methods MUST describe the security issues that relate to each particular method.

6. Acknowledges

Duncan Mills and Miguel A. Garcia-Martin supported the idea of 1 to n MESSAGEs. Jon Peterson and Dean Willis provided useful comments.

7. References

7.1 Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

7.2 Informational References

[3] Camarillo, G., "Providing a Session Initiation Protocol (SIP) Application Server with a List of URIs", draft-camarillo-sipping-uri-list-01 (work in progress), February 2004.

[4] Roach, A., Rosenberg, J. and B. Campbell, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource

Lists", draft-ietf-simple-event-list-04 (work in progress), June 2003.

[5] Rosenberg, J., "Advanced Instant Messaging Requirements for the Session Initiation Protocol (SIP)", draft-rosenberg-simple-messaging-requirements-01 (work in progress), February 2004.

[6] Rosenberg, J., "An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Presence Lists", draft-ietf-simple-xcap-list-usage-02 (work in progress), February 2004.

[7] Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-02 (work in progress), July 2003.

[8] Rosenberg, J. and H. Schulzrinne, "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-03 (work in progress), February 2004.

[9] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-02 (work in progress), February 2004.

[10] Bradner, S., "A Proposal for an MOU-Based ICANN Protocol Support Organization", RFC 2690, September 1999.

Author's Address

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.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.

Sipping
Internet-Draft
Expires: January 9, 2005

P. Kyzivat
Cisco Systems, Inc.

July 11, 2004

Reg Event Package Extension for GRUUs
draft-kyzivat-sipping-gruu-reg-event-00

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 9, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This draft defines an extension to RFC 3680 [1] for representing the GRUU associated with a Contact.

1. Introduction

The proposal to add GRUU support to the REGISTER message [2] introduces another element of state to the registrar. Subscribers to the registration event package [1] will sometimes have need for the

new state.

For example, the Welcome Notices example in [1] will only operate correctly if the contact address in the reg event notification is reachable by the sender of the welcome notice. When the registering device is using the gruu extension, it is likely that the registered contact address will not be globally addressable, and the gruu should be used as the target address for the MESSAGE.

The reg event package has provision for including extension elements within the <contact> element. This draft proposes a new element that may be used in that context to deliver the GRUU corresponding to the contact.

2. Description

A new element (<gruu>) is defined which simply contains the GRUU.

A notifier for the "reg" event package SHOULD include this element when a contact has an Instance ID and a GRUU is associated with the combination of the AOR and the Instance ID. When used, the <gruu> element MUST be used within the <contact> element.

3. Example

The following is an example registration information document:

```
   &lt;?xml version="1.0"?&gt;
       &lt;reginfo xmlns="urn:ietf:params:xml:ns:reginfo"
           xmlns:gr="urn:ietf:params:xml:ns:gruu"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           version="0" state="full"&gt;
         &lt;registration aor="sip:user@example.com" id="as9"
             state="active"&gt;
           &lt;contact id="76" state="active" event="registered"
              duration-registered="7322"
              q="0.8"&gt;
                &lt;uri&gt;sip:user@192.0.2.1&lt;/uri&gt;
                &lt;unknown-param name="+sip.instance"&gt;
                   "&lt;urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6&gt;"
                &lt;/unknown-param&gt;
                &lt;gr:gruu&gt;sip:hha9s8d=-999a@example.com&lt;/gruu&gt;
           &lt;/contact&gt;
         &lt;/registration&gt;
       &lt;/reginfo&gt;
```

4. XML Schema Definition

An gruu document is an XML document that MUST be well-formed and SHOULD be valid. Gruu documents MUST be based on XML 1.0 and MUST be encoded using UTF-8. This specification makes use of XML namespaces for identifying gruu documents. The namespace URI for elements defined for this purpose is a URN, using the namespace identifier 'ietf'. This URN is:

urn:ietf:params:xml:ns:gruu

```
&lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;xs:schema targetNamespace="urn:ietf:params:xml:ns:gruu"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="urn:ietf:params:xml:ns:gruu"&gt;
  &lt;xs:element name="gruu" type="xs:anyURI"/&gt;
&lt;/xs:schema&gt;
```

5. IANA Considerations

TBD.

6. Security Considerations

Security considerations for the registration event package is discussed in RFC 3680 [1], and those considerations apply here.

The addition of gruu information does not impact security negatively because the gruu is less sensitive than the contact URI itself.

7. Acknowledgements

The author would like to thank Jonathan Rosenberg for encouraging this draft.

8. References

8.1 Normative References

[1] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations", RFC 3680, March 2004.

[2] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-02 (work in progress), July 2004.

8.2 Informative References

[3] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

Author's Address

Paul H. Kyzivat
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA 01719
USA

EMail: pkyzivat@cisco.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 procedures with respect to rights in RFC 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.

Pros and Cons of allowing SIP Intermediaries to add MIME bodies
draft-mahy-sipping-body-add-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 30, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

The SIPPING Working Group has developed requirements for session policy (including bandwidth and codec restrictions, logging, and middlebox traversal), request history, and identity. This document discusses the pros and cons of allowing intermediaries to add SIP message bodies to address these requirements. It is intended to provoke serious discussion rather than as a complete proposal.

Table of Contents

1. Conventions

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 [2].

2. Introduction

Certain classes of applications the SIP and SIPPING WGs are considering ([9], [10], [11], [12]), lend themselves casually to an approach where intermediaries can inspect, add, or modify bodies. Unfortunately, this practice as currently implemented is completely incompatible with the S/MIME [3] end-to-end security mechanisms specified in the core SIP specification (RFC 3261 [1]), and consequently an explicit violation of the spec. The SIP community is at an impasse regarding how these classes of feature need to be implemented. This document attempts to present an overview of the two major proposals for moving forward.

One proposal suggests that body additions (as opposed to modifications) can be done safely by SIP intermediaries if these bodies are optional in nature and if certain restrictions are placed on which intermediaries are allowed to add bodies and under what circumstances. This proposal would require a relaxation of one sentence in the SIP specification and would effectively enable a generic mechanism which could be used for a variety of applications. This mechanism would not interfere with user agents which do end-to-end security directly. Intermediaries which could add bodies could sign or encrypt these as the product of a specific intermediary. The receiving user agent would be responsible for verifying the validity and trustworthiness of each body part.

Another proposal suggests that allowing intermediaries to add bodies introduces unneeded complexity and a handful of other undesirable properties. These undesirable properties could be avoided by addressing each of the requirements individually while carefully limiting the scope of some of these applications. In addition, this proposal accommodates a model where a new intermediary role called an Authentication Service which has a direct TLS [4] connection and a specific trust relationship with one of the user agents could make change to bodies on behalf of that user agent if also performing end-to-end security operations on its behalf.

In addition to supporting the required applications in the presence of true end-to-end security, it is highly desirable to support a mechanism that allows specific intermediaries to safely sign and verify and possibly encrypt and decrypt requests and responses on behalf of user agents which have not implemented S/MIME. This allows

for a migration path from existing implementations to a completely end-to-end environment in a safe manner.

3. Topologies

An explicit policy fetch allows user agents to fetch a policy document directly from their intermediaries, for example using the approach described in [14] and [13]. This significantly reduces, but does not completely eliminate the need for policy "corrections" by specific intermediaries for specific sessions. The remainder of this section assumes that available policy information available in the local domain has already been exhausted. Note that through configuration or prior negotiation, Alice and atlanta.com probably have few policy conflicts, and Bob and biloxi.com probably have few policy conflicts. The bulk of policy conflicts are likely to be between Alice or atlanta.com and Bob or biloxi.com.

This section explores the possible paths that a request can take from sip:alice@phone2.atlanta.com to sip:bob@pc1.biloxi.com and the policy implications.

Full Redirect Model: This topology results in Alice sending a request to sip:bob@biloxi.com, which redirects her request to bob@pc1.biloxi.com. Alice opens a new connection directly to pc1.biloxi.com and sends her request directly with no intermediate proxies . There is no opportunity for either atlanta.com or biloxi.com to enforce session policy here at all, since neither is involved in further signaling.

```
INVITE sip:bob@biloxi.com SIP/2.0

[biloxi.com redirects]
SIP/2.0 302 Moved
Contact: <sip:bob@pc1.biloxi.com>

[Alice retries request to new target URI]
INVITE sip:bob@pc1.biloxi.com SIP/2.0
```

Triangle Signaling Model: Alice opens a connection to biloxi.com which routes Alice’s request to bob@pc1.biloxi.com. This offers an opportunity for biloxi.com to issue a repairable error response which Alice could fix and retry. This is the most elegant toplogy because it has the simplest security characteristics. Unfortunately this model does not allow atlanta.com to influence requests from Alice. Many organizations require policy influence over requests which originate within their networks.

INVITE sip:bob@biloxi.com SIP/2.0

[biloxi.com retargets]
INVITE sip:bob@pc1.biloxi.com SIP/2.0

Trapezoid Signaling Model: Alice routes its request to bob@biloxi.com through atlanta.com. Then biloxi.com retargets the requests and forwards it to bob@pc1.biloxi.com. This model allows both atlanta.com and biloxi.com to influence policy on new sessions. There are still variations of how atlanta.com and biloxi.com can influence the session.

INVITE sip:bob@biloxi.com SIP/2.0
Route: sip:atlanta.com;lr

[atlanta.com forwards the request to biloxi.com]
INVITE sip:bob@biloxi.com SIP/2.0

[biloxi.com retargets]
INVITE sip:bob@pc1.biloxi.com SIP/2.0

One way to allow proxies to influence policy in the trapezoid model causes an extra round trip to allow Alice to "consent" to each proposed policy change. For example, atlanta.com could issue a repairable error response to influence a new request, and then biloxi.com could likewise issue a repairable error response to add its policy requirements. This model results in many messages and can result in significant additional delay due to extra round trips. In addition, information which is potentially private between biloxi.com and Bob is sent to Alice. Also, Alice may be asked to forward opaque or encrypted data from an intermediary with whom Alice has no trust relationship. It hard to imagine how Alice could decide on what basis to "consent" to include such content.

Alice -> atlanta.com

[atlanta.com asks Alice to comply with specific policy]
Alice -> atlanta.com -> biloxi.com

[biloxi.com asks Alice to comply with specific policy
or forward opaque data to Bob]
Alice -> atlanta.com -> biloxi.com -> bob@pc1.biloxi.com

Alternatively, many existing deployments "piggyback" extra information at atlanta.com and biloxi.com (or modify the MIME [5] content). In addition to expressly violating RFC 3261 [1] and breaking any end-to-end security used by Alice and Bob, this model can cause Alice or Bob to receive MIME bodies with Content-Types

which they don't understand (This is known as the "415" problem after the 415 response code). Imagine Alice sends a requests including only the text/foo MIME type, but receives a 415 Unacceptable response which includes text/foo as an acceptable MIME type. Alice has no information about what happened (Bob rejected the text/bar MIME type inserted by atlanta.com), and cannot do anything to repair the "error".

The compromise approach described in the next section allows atlanta.com to "challenge" Alice with repairable error responses to comply with atlanta's policies, while biloxi.com can add a message body intended for consumption by Bob. This may be a technically workable solution, but requires complex MIME and authorization processing by intermediaries that participate in policy. This approach would still require a relaxation of Section 16.6, Step 1 of RFC 3261 [1].

Alice -> atlanta.com

[atlanta.com asks Alice to comply with specific policy]
Alice -> atlanta.com -> biloxi.com

[biloxi adds its policy requests to the request]
-> Bob

Pentagram Signaling Model: In this model, extra intermediaries who are not directly associated with either Alice or Bob are included. This model is to be avoided as it dramatically increases the complexity of the security required.

Alice -- atlanta.com -- provider.net -- biloxi.com -- Bob

4. Overview of the body addition proposal

Of prime importance to the body addition proposal is insuring that the mechanism can be added in a backwards compatible way. To facilitate backward compatibility, the body addition proposal introduces a new option-tag called "repack" which indicates that a user agent supports multipart MIME [6] and allows bodies to be addressed to and from intermediaries. User Agents include this token in a Supported header when registering along with an Accept header with all the MIME types the User Agent supports.

When a User Agent supports body repacking, we assume that the wrapping of the outermost MIME type in the SIP body is not relevant for the authentication purposes. Each of the MIME parts inside the outermost part can stand alone as a separate message. Each of these

MIME parts MUST have a Content-Disposition MIME header. If the MIME part is sent to or from an intermediary (instead of the original UAC or the final UAS), the Content-Disposition header MUST contain a src or dest parameter indicating the source or destination of the request.

If the UAC needs to include some content for a specific intermediary, it indicates this by adding a content parameter to the Route header field value which corresponds to the target intermediary. The content parameter contains a content ID [7] which is referenced in the appropriate body. (For illustrative purposes, a band of asterisks (*****) surrounds content that would actually be signed or encrypted using S/MIME).

```
INVITE sip:bob@biloxi.example.com
From: <sip:alice@atlanta.example.com>
To: <sip:bob@biloxi.example.com>
Route: <sip:atlanta.example.com;lr>;content="lki290s8"
...
Content-Type: multipart/mixed;boundary=bar

--bar
Content-ID: <lki290s8>
Content-Disposition: policy ;handling=optional ;dest="sip:atlanta.example.com"
*****
* ...
*****
--bar
Content-Disposition: session ;handling=required
*****
* Content-Type: application/sdp
* ...
*****
--bar
```

When an intermediary operating on the UACs behalf requires additional information in a request it needs to send a repairable error response asking for the appropriate additional information. We can define a new response code for this, for example "497 Policy Error". However, the UAC and intermediaries operating on the UACs behalf are expected to be well matched, for example mutually configured using session independent policies, so this extra round trip should not happen very often.

When an intermediary operating on behalf of the UAS needs to include additional information about the request, it can add a body part to the message if it knows that the UAS supports the repack option and that any required body types that were added are acceptable to the

UAS. In most cases, the UAS registered with the repack option-tag in a Supported header or is administratively configured to known that the UAS supports the extension. In addition, the UAS proxy can include any MIME types as long as the handling parameter (in the Content-Disposition header) indicates the body part is optional. In addition, if the intermediary is collocated with the registrar for the UAS, the intermediary can observe the MIME types listed in the Accept header and send these even as "required" body parts. Any body parts added by the intermediary need to have a src parameter which corresponds the SIP URI of the intermediary that added the body part. In addition, these MIME parts MUST be signed using S/MIME using the key from a certificate which contains a SubjectAltName field which exactly matches the SIP URI in the src parameter.

```
Content-Disposition: policy ;handling=optional ;src="sip:biloxi.example.com"
*****
* Content-Type: application/sip-session-policy+xml
* ...
*****
```

When a UAC receives a request, it MUST examine the src parameter for each body type that it understands. If any of the body parts are signed it then must discard body parts from untrusted sources, or improperly signed body parts. The UAC can then clearly distinguish the body parts which were signed by the UAC from the body parts that were signed by the a proxy operating on behalf of the UAS.

When the UAS sends a response, intermediaries operating on behalf of the UAS can examine the response and forward the response along. Typically the response will cooperate with the policies that were just sent in the request, but if not, the intermediary can send a 500 Server Error response to the request and drop the illegal response it received from the UAS. Intermediaries can similarly add body parts to the response as long as the UAC indicated support for the repack option-tag and all "required" MIME types are acceptable to the UAC. Finally, when the UAC receives the response, it MUST examine the src parameter for each body type that it understands, discard untrusted or improperly signed body parts and act on body parts sent by the UAS differently from body parts added by its intermediary.

This proposal addresses the three most serious technical concerns with adding bodies. The proposal is backward safe and can operate even if only one side supports the extension. It is impossible for the UAC to receive a 415 Not Acceptable response due to content inserted by an intermediary. The User Agents can distinguish which body parts were sent by the other User Agent and which were added by an intermediary.

Unfortunately the proposal requires very sophisticated MIME parsing and verification/generation of multiple S/MIME signatures per message on both User Agents and intermediaries which decide to add bodies. This requires that UACs either sign all bodies, no bodies, or that they trust an appropriate service to do so (and that the protocol support necessary for this is available). On first glance, it may also seem to increase message size and processing time, however initial analysis does not suggest any significant difference between this approach and any other proposals in this regard. Note also, that this approach opens up opportunities for intermediaries to abuse this functionality for so-called "middle-to-middle" communications which can introduce a significant burden on other SIP intermediaries and the infrastructure of the Internet.

Finally, this approach can be modified slightly to allow a 3rd party user agent to sign, verify, encrypt, and decrypt SIP messages on behalf of a user agent which does not support end-to-end security. This SIP node would keep credentials for the address-of-record of the user agent and apply these to each of its messages. It could handle all the authorization and verification duties (for example, throwing away bodies inserted by malicious intemediaries) normally required of user agents under this proposal.

5. Applications with and without body modification

5.1 Logging

This application [10] requires an intermediary to inspect SIP message bodies. This can be session descriptions [18] which reference specific streams, or in the case of the MESSAGE method [22], actual content. If this session description or content is encrypted, either the logging service needs to receive a copy of the Content Encryption Key or it needs to receive another copy of the message.

It is clear that if Alice wants to provide a copy of Content Encryption Key to her logging proxy she can, but less clear how she can (directly or indirectly) provide this information to Bob's logging proxy. Bob could provide this information to its proxy, but this requires that either Bob's proxy ask for this information (and that Alice provide it) or that Bob provide the Content Encryption Key to his proxy in a way that is easy to correlate.

For this application, it is not necessary for an intermediary to ever add its own body (commonly called end-to-middle [15] security or e2m). Addressing some bodies from a user agent to an intermediary instead of the other user agent could be used here, but this application could be accomodated nearly as easily without addressing bodies at intermediaries.

5.2 Session codec / bandwidth policy checking

This application requires an intermediary to inspect session descriptions, but does not require them to be changed. This is problematic if the session description is encrypted however, especially if the session description contains keying information [24] which Alice or Bob don't want to be provided to an intermediary and is not otherwise required.

Directing a copy of a portion of the session description at an intermediary (e2m) could mitigate the privacy lost here, but does not require body addition.

5.3 Midcom-style firewall traversal

Like the previous application, a firewall traversal intermediary (ex: using the MIDCOM architecture [19]) needs access to the transport protocols, IP addresses, and ports in use for each m-line. Again, if the session description is encrypted and contains sensitive keying material, it would be desirable to provide an additional copy of this information in another body using e2m. No body additions by intermediaries are required for this application either.

5.4 NAT traversal (including v4/v6 translators)

NAT [20] traversal using protocols such as STUN [8] and ICE [25] would not normally require body modification, addition, or even inspection. (An intermediary might need to provide an address of a STUN server for example.) NAT traversal using a MIDCOM-style approach however introduces a tremendous amount of complexity.

This application is fairly complex with the body modification proposal (a specific proposal is described in the next paragraph, which does), and even more challenging when body modifications are not permitted. However, it may be prudent for the SIP community to completely reject this as a valid application of the SIP session policy mechanism when superior mechanisms for NAT traversal are available.

[Description of MIDCOM-style NAT traversal with body addition approach]

5.5 3rd-party Asserted Identity

This application involves an intermediary providing an assertion of the identity of the sender of the message. A proposal which describes this concept using a body (in this case an authenticated identity body) is described in

<http://www.softarmor.com/wgdb/docs/draft-ietf-sip-identity-00.txt>. A proposal which describes this concept using a header is [11]. Note that the auth-id [16] body could be replaced with a different body to allow unambiguous use in both requests and responses.

End-to-end identity could be provided in such a way to provide a secure binding between the original Request-URI and a Contact header provided. When used in a body, this would unfortunately require a new Identity header anytime a Contact header changes (for example when transitioning from a 2-party call to a SIP conference [26]).

5.6 Request History

This application involves an intermediary providing an assertion that a request was retargetted. Request history using body addition [12] is extremely natural. An auth-id body is provided for every retargetting signed by the proxy performing that retargetting. This provides an alternative way of binding an original Request-URI with a provided Contact header.

History without body addition could be accomplished in one of three ways. The Request-History header field value itself could contain a cryptographic object similar to the current Identity proposal. The Request-History service could be restricted so it can only be provided by a server which also provides the Identity service (the most common cases), Finally, request history could be provided as a P-Header [21] only for use in certain administrative domains using a technique similar to RFC 3325 [23] (P-Asserted-Identity) that requires specific trust relationships.

5.7 3rd Party Authentication Service

This service signs, verifies, encrypts, and decrypts on behalf of a user agent which cannot perform these functions itself. A third party which performs these functions most definitely needs to inspect and add MIME bodies. This third part however would have credentials used on behalf of the user, and would presumably be reachable directly over a secure channel (for example over a TLS connection). This application is easily implemented using the body addition proposal. If Alice needed a new request signed or encrypted she would need to send her request to this server, which would return her signed or encrypted content. She would then resend the request. Bob's service could add a MIME body with the decrypted and verified contents, and also encrpyt and/or sign Bob's response.

As an alternative to the body addition proposal, you could relax the body modification requirement just for this specific application which would be defined as a new SIP role with specific normative

behavior.

In either case, such a service could be collocated with a SIP Credential Service [17] or an <http://www.employees.org/~fluffy/ietf/draft-jennings-sipping-certs-02.html> .

6. Security Considerations

Much to talk about here.

7. IANA Considerations

8. Acknowledgments

Thanks to Jon Peterson and Cullen Jennings for a hearty discussion.

9. References

9.1 Normative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

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

[4] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[5] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[6] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[7] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, August 1998.

[8] Rosenberg, J., Weinberger, J., Huitema, C. and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through

Network Address Translators (NATs)", RFC 3489, March 2003.

[9] Rosenberg, J., "Requirements for Session Policy for the Session Initiation Protocol (SIP)", draft-ietf-sipping-session-policy-req-01 (work in progress), February 2004.

[10] Ono, K. and S. Tachimoto, "Requirements for End-to-middle Security for the Session Initiation Protocol (SIP)", draft-ietf-sipping-e2m-sec-reqs-03 (work in progress), July 2004.

[11] Peterson, J., "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-02 (work in progress), May 2004.

[12] Barnes, M., "An Extension to the Session Initiation Protocol for Request History Information", draft-ietf-sip-history-info-02 (work in progress), February 2004.

[13] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-03 (work in progress), May 2004.

[14] Hilt, V., "Session-Independent Policies for the Session Initiation Protocol (SIP)", draft-hilt-sipping-session-indep-policy-01 (work in progress), May 2004.

[15] Ono, K. and S. Tachimoto, "End-to-middle security in the Session Initiation Protocol(SIP)", draft-ono-sipping-end2middle-security-02 (work in progress), May 2004.

[16] Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-03 (work in progress), May 2004.

[17] Jennings, C., "Certificate Management Service for SIP", draft-jennings-sipping-certs-03 (work in progress), May 2004.

9.2 Informational References

[18] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[19] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A. and A. Rayhan, "Middlebox communication architecture and framework",

RFC 3303, August 2002.

[20] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[21] Mankin, A., Bradner, S., Mahy, R., Willis, D., Ott, J. and B. Rosen, "Change Process for the Session Initiation Protocol (SIP)", BCP 67, RFC 3427, December 2002.

[22] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[23] Jennings, C., Peterson, J. and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002.

[24] Andreasen, F., Baugher, M. and D. Wing, "Session Description Protocol Security Descriptions for Media Streams", draft-ietf-mmusic-sdescriptions-06 (work in progress), July 2004.

[25] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols", draft-ietf-mmusic-ice-01 (work in progress), February 2004.

[26] Johnston, A. and O. Levin, "Session Initiation Protocol Call Control - Conferencing for User Agents", draft-ietf-sipping-cc-conferencing-03 (work in progress), February 2004.

Author's Address

Rohan Mahy
Cisco Systems, Inc.
5617 Scotts Valley Drive, Suite 200
Scotts Valley, CA 95066
USA

EMail: rohan@cisco.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 procedures with respect to rights in RFC 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.

Network Working Group A. Niemi
Internet-Draft Nokia
Expires: January 17, 2005 July 19, 2004

Session Initiation Protocol (SIP) Event Notification Throttles
draft-niemi-sipping-event-throttle-01

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 17, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This memo specifies a throttle mechanism for limiting the rate of Session Initiation Protocol (SIP) event notifications. This mechanism can be applied in subscriptions to all SIP event packages.

Table of Contents

1. Introduction

The SIP events framework [1] defines a generic framework for subscriptions to and notifications of events related to SIP systems. This framework defines the methods SUBSCRIBE and NOTIFY, and introduces the concept of an event package, which is a concrete application of the SIP events framework to a particular class of events.

One of the things the SIP events framework mandates is that each event package specification defines an absolute maximum on the rate at which notifications are allowed to be generated by a single notifier. Such a limit is provided in order to reduce network congestion.

All of the existing event package specifications include a maximum notification rate recommendation, ranging from once in every five seconds [4], [5], [6] to once per second [7].

Per the SIP events framework, each event package specification is also allowed to define additional throttle mechanisms which allow the subscriber to further limit the rate of event notification. So far none of the event package specifications have defined such a mechanism.

This document defines an extension to the SIP events framework that allows a subscriber to set a throttle to event notifications generated by the notifier. The requirements and model for generic event throttles are further discussed in Section 3. A throttle is simply a timer value that indicates the minimum time period allowed between two notifications. As a result of this throttle, a compliant notifier will limit the rate at which it generates notifications.

2. Definitions and Document Conventions

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 [2] and indicate requirement levels for compliant implementations.

> Indented passages such as this one are used in this document to provide additional information and clarifying text. They do not contain normative protocol behavior.

3. Overview

3.1 Use Case

There are many applications that potentially would make use of a throttle mechanism. This chapter only illustrates one possible use case, in which a device uses the event throttling mechanism to limit the amount of traffic it may receive.

3.1.1 Pre-conditions

A presence application in Lisa's device contains a list of 100 presentities. In order to decrease the processing and network load of watching 100 presentities, Lisa's presence application has included an event throttle to each of the subscriptions, to limit the maximum rate at which notifications are to be generated to once per 20 seconds.

Heikki is one of the presentities Lisa is watching. Heikki's presence agent conforms to the throttling policy requested by Lisa's presence application. The event package includes only full-state notifications.

3.1.2 Normal Flow

- Heikki publishes a presence status of "red", which results in a presence notification to be sent to Lisa.
- In 10 seconds, Heikki publishes a presence status of "blue". As the throttling policy set by Lisa only allows the presence agent to generate notifications at a maximum of once per 20 seconds, the notification is put on hold.
- After another 10 seconds, the notification is allowed to be sent to Lisa.
- Lisa receives a presence update conforming to the set throttling policy.

3.1.3 Alternative Flow I

- Heikki publishes a presence status of "red", which results in a presence notification to be sent to Lisa.
- In 10 seconds, Heikki publishes a presence status of "blue". As the throttling policy set by Lisa only allows the presence agent to generate notifications at a maximum rate of once per 20 seconds, the notification is put on hold.

- After another 5 seconds, Heikki publishes a presence status of "green". The resulting notification is not conformant to the throttling policy set by Lisa and is therefore put on hold, replacing the earlier queued notification (the one containing the status of "blue").

- Lisa receives a presence update conforming to the set throttling policy and containing the "green" status.

3.1.4 ALternative Flow II

Instead of full state, the notifications now contain partial-state.

- Heikki publishes a presence status of "red", which results in a presence notification to be sent to Lisa.

- In 10 seconds, Heikki publishes a changed presence status of "blue". As the throttling policy set by Lisa only allows the presence agent to generate notifications at a maximum rate of once per 20 seconds, the notification is put on hold.

- After another 5 seconds, Heikki publishes an additional presence status of "bitter". The resulting notification is not conformant to the throttling policy set by Lisa and is therefore put on hold. Since there already exists a queued notification (that of the "blue" status"), the notifier merges the two notifications into a single notification (containing both "blue" and "bitter" statuses).

- Lisa receives a presence update conforming to the set throttling policy and containing the "blue" and "bitter" status.

3.1.5 Post-conditions

Lisa receives notifications of Heikki's presence at a maximum of once per 20 seconds. Only newest notifications containing full-state are ever sent to Lisa. With partial-notifications, the notifier merges the states of all notifications generated within a single 20 second period.

3.2 Requirements

REQ1: The subscriber MUST be able to set using a throttle mechanism the minimum time period between two notifications in a specific subscription.

REQ2: The subscriber MUST be able to indicate that it requires the notifier to comply with the suggested throttling policy in a specific subscription.

REQ3: The notifier MUST be able to indicate that it does not support the use of a throttle mechanism in the subscription.

REQ4: It MUST be possible to use the throttle mechanism in subscriptions to all events.

REQ5: It MUST be possible to use the throttle mechanism together with any event filtering mechanism.

REQ6: The notifier MUST be allowed to use a throttling policy in which the minimum time period between two notifications is longer than the one given by the subscriber.

For example, due to congestion reasons, local policy at the notifier could temporarily dictate a throttling policy that in effect increases the subscriber-configured minimum time period between two notifications.

REQ7: The throttle mechanism MUST provide a reasonable resolution for setting the minimum period between two notifications. At a minimum, the throttling mechanism MUST include discussion of the situation resulting from a minimum time period which exceeds the subscription duration, and SHOULD provide mechanisms for avoiding this situation.

REQ8: A throttle mechanism MUST allow for the application of authentication and integrity protection mechanisms to subscriptions invoking that mechanism.

Note that Section 7 contains further discussion on the security implications of the throttle mechanism.

3.3 Event Throttle Model

Using notations from traffic theory, we can model the notifier as a statistical multiplexer with an input rate of Ci (i = 1,...,n), and an output rate of C <= C1 + ... + Cn. Typically, the statistical multiplexer is lossy, with a finite buffer size. The loss probability of the statistical multiplexer can be decreased by enlarging this buffer. Figure 1 illustrates the model.

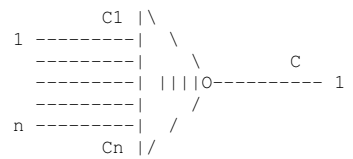


Figure 1: Notifier modeled as a statistical multiplexer

The output connection has a default rate that is generally dictated by each individual event package. The rate can also be set using the throttle mechanism described in this document. A notifer that accepts a subscriber-defined throttle, adjusts its output rate accordingly.

There is typically only a single input connection, characterized by the event package, and consisting of a stream of event notifications. In general, applying a throttle MUST NOT alter the properties of the buffer. I.e., the event notifications are expected to deliver consistent information even when throttled.

In practice, there are only two viable buffer policies for SIP event notifiers:

Full-state: Last one in is sent out, and all others in the buffer are discarded. This policy applies to those event packages that carry full-state notifications.

Partial-state: The states of buffered notifications are merged, and the resulting notification is sent out. This policy applies to those event packages that carry partial-state notifications.

3.4 Basic Operation

A subscriber that wants to limit the rate of event notification in a specific subscription does so by suggesting a throttle as part of the SUBSCRIBE message. The throttle indicating the minimum time allowed between two notifications in a subscription is given as an Event header parameter in the SUBSCRIBE request.

The subscriber also indicates that it requires the throttle to be applied to the subscription. This is done using the SIP option-tag mechanism, by insisting that the notifier applies the event throttle extension when processing the request. A notifier that does not support the event throttle extension will reject the subscription.

A notifier that supports the throttle mechanism will comply with value given in the throttle, and adjust its rate of notification accordingly.

Throttled notifications will have exactly the same properties as the un-throttled ones, with the exception that they will not be generated more frequent than what the throttle allows.

4. Operation of Event Throttles

4.1 Negotiating the Use of Throttle

This specification uses the SIP option-tag mechanism for negotiating use of the throttle mechanism. Use of the "Require" header field and the 420 (Bad Extension) are according to SIP [3].

A subscriber that wishes to apply a throttle to notifications in a subscription insists that the notifier applies this throttle by including an "event-throttle" option-tag to the Require header field of the SUBSCRIBE request.

A notifier that does not understand the event-throttle extension, will respond with a 420 (Bad Extension) response. Otherwise, the throttle is processed by the notifier, and the notification rate is adjusted accordingly.

4.2 Setting the Throttle

4.2.1 Subscriber Behavior

In general, the way in which a subscriber generates SUBSCRIBE requests and processes NOTIFY requests is according to RFC 3265 [1].

A subscriber that wishes to throttle the notifications in a subscription includes a "throttle" Event header parameter in the SUBSCRIBE request, indicating in seconds the throttle value. The value of this parameter is an integral number of seconds in decimal.

In case the notifier does not support the "event-throttle" extension, the subscriber SHOULD retry the subscription without that extension.

> In this case the subscriber can resort to other means of limiting the notification rate. For example, instead of a subscription, it can fetch or poll the event state.

4.2.2 Notifier Behavior

In general, the way in which a notifier processes SUBSCRIBE requests and generates NOTIFY requests is according to RFC 3265 [1].

A notifier that supports the "event-throttle" extension extracts the value of the "throttle" Event header parameter, and uses it as the minimum time allowed between two notifications.

A notifier MUST include the selected throttle value in a "throttle" parameter to the Subscription-State header field of the NOTIFY requests sent to the subscriber.

A compliant notifier MUST NOT generate notifications more frequent than what the throttle allows for, except when generating the notification either upon receipt of a SUBSCRIBE request (the first notification) or upon termination of the subscription (the last notification).

As specified in RFC 3261 [3] a notifier that supports event throttles SHOULD advertise its support by including the "event-throttle" option-tag in the Supported header field of a response to an OPTIONS request.

4.3 Selecting the Throttle Interval

Special care needs to be taken when selecting the throttle value. Using the throttle syntax it is possible to insist both very short and very long throttles to be applied to the subscription. For example, a throttle could potentially set a minimum time value between notifications that exceeds the subscription expiration value. Such a configuration would effectively quench the notifier, resulting in exactly two notifications to be generated.

OPEN ISSUE: Should we give recommendations to reasonable throttle resolutions, or define what behavior to exhibit if an unreasonable throttle value is given to the notifier?

5. Syntax

This section describes the syntax extensions required for watcherinfo history. Note that the formal syntax definitions described in this section are expressed in the Augmented BNF format used in SIP [3], and contain references to elements defined therein.

5.1 "event-throttle" SIP Option Tag

The "event-throttle" SIP option-tag is added to the "option-tag" definition in the SIP grammar. Usage of this option-tag is defined in Section 4.1.

5.2 "throttle" Header Parameter

The "throttle" header parameter is added to the "generic-param" definition in the SIP grammar. Usage of this Event header parameter is described in section Section 4.2.

5.3 Augmented BNF Definitions

This section describes the Augmented BNF definitions for the new syntax element. The notation is as used in SIP [3] and the documents to which is refers.

```
generic-param  =  throttle-param / token [ EQUAL gen-value ]
throttle-param =  "throttle" EQUAL delta-seconds
option-tag     =  throttle-tag / token
throttle-tag   =  "event-throttle"
```

6. IANA Considerations

TBD: New SIP option tag (event-throttle), and possibly new header parameter (throttle) need to be registered with IANA.

7. Security Considerations

Naturally, the security considerations listed in SIP events [1], which the throttle mechanism extends, apply in entirety. In particular, authentication and message integrity SHOULD be applied to subscriptions with the event-throttle extension.

8. Acknowledgements

Thanks to Pekka Pessi, Dean Willis, Eric Burger, Alex Audu , Alexander Milinski, and the SIPPING WG for support and review of this work.

9. References

9.1 Normative References

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

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

[3] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

9.2 Informative References

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

[5] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations", draft-ietf-sipping-reg-event-00 (work in progress), October 2002.

[6] 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.

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

Author's Address

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

Phone: +358 50 389 1644
EMail: aki.niemi@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 procedures with respect to rights in RFC 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.

SIPPING K. Ono
Internet-Draft S. Tachimoto
Expires: November 15, 2004 NTT Corporation
May 17, 2004

End-to-middle Security in the Session Initiation Protocol(SIP)
draft-ono-sipping-end2middle-security-02

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 15, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

Some services provided the intermediaries depend on the ability to inspect the message bodies in the Session Initiation Protocol (SIP). When sensitive information is included in them, a SIP User Agent needs to protect it from all intermediaries except the certain selected intermediaries. This document proposes a mechanism for securing information passed between an end user and a selected intermediary using S/MIME. This also proposes a mechanism for the discovery of an intermediary that needs to inspect an S/MIME-secured message body.

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 [1].

Table of Contents

1. Introduction

When a SIP [2] UA requires services that are provided by intermediaries depending on the message bodies in request/response messages, end-to-end confidentiality will currently have to be disabled to take advantage of the services. This problem is pointed out in Section 23 of [2]. Since such an intermediary is not always adjacent to the UA, this situation requires security between the UA and the intermediary for the message bodies. We call this "end-to-middle security", where by "end" we mean a UA and by "middle" we mean a specific intermediary, typically a proxy server.

This document describes proposed mechanisms to provide data confidentiality and integrity for end-to-middle security to meet the requirements discussed in [3]. Since the major requirement is to have little impact on the standardized end-to-end security mechanisms, the proposed mechanisms are based on S/MIME [4]. The mechanisms consist of generating S/MIME CMS [5] data and indicating target message body for a selected proxy server. In addition, it also includes a mechanism for the discovery of selected proxy servers.

2. Generating S/MIME CMS Data

For end-to-middle confidentiality, a UAC MUST be able to generate S/MIME CMS EnvelopedData, whose recipients are specified in the "recipientInfos" field. The structure of the S/MIME CMS EnvelopedData contains data encrypted with a content-encryption-key (CEK) and the CEK encrypted with different key-encryption-keys (KEKs), one for each recipient as specified in [5]. The KEKs are the public keys of each recipient or the shared keys between the UAC and each recipient.

If the data is encrypted only for a selected proxy server, the recipients contain only the proxy server. If there is encrypted data for destined for different proxy servers, the recipient list of each encrypted piece of data will contain the targeted proxy servers. The UAC MUST generate a multipart MIME body to contain the encrypted data. When the message body including the encrypted data is transmitted to a UAS, the UAS will be unable to decrypt it. The UAC MUST set the value "optional" in the handling parameter of the Content-Disposition MIME header for the message body in order to avoid causing unneeded error condition in the UAS.

Open Issue: Is it necessary that a multipart MIME body contains the handling parameter of Content-Disposition header? How should it be set when the multipart MIME body contains a body with the value "required" and another body with the value "optional"?

If the encrypted data is meant to be shared with the UAS and selected proxy servers, the recipients SHOULD be addressed to the UAS and

selected proxy servers. The UAS SHOULD decrypt the message body including the encrypted data. The UAC SHOULD set the value "required" in the handling parameter of the Content-Disposition MIME header for the message body. If the handling parameter is not set, the default behavior is the same as setting the value "required" as specified in [2]

If an encrypted piece of data is destined for a selected proxy server and another encrypted data is for the UAS, the recipient of each encrypted data is each entity. In this case, the UAC MUST generate them part of a multipart MIME body.

For example, UAs use this method when keying materials, such keys for use by Secure RTP (SRTP), are included in the SDP[6]. One CMS EnvelopedData body contains SDP that includes keying materials of an SRTP stream only for the UA. The other EnvelopedData body contains an SDP that does not include the keying materials of an SRTP stream only for a selected proxy server that needs to view SDP (i.e.: for a firewall traversal service).

For end-to-end data integrity, UAs use S/MIME CMS SignedData body that can be validated by any entity. Therefore no new CMS SignedData generating mechanism is required for end-to-middle data integrity.

Note: Even when the handling parameter is set to the value "optional", the UAS SHOULD validate the signature of whole MIME body, since Content-Disposition might be modified by a malicious entity.

3. Indicating the Target Content

UAs needs a way to indicate the target of the content in order that a proxy server can easily determine whether to process S/MIME bodies and if so, which one. The UA SHOULD set a new "Content-Target" MIME header to label the target message body for a selected proxy server. When UAs label the encrypted data, the UA SHOULD set the "Content-Target" MIME header of the S/MIME CMS EnvelopedData. When UAs label the data with signature, the UA SHOULD set the "Content-Target" MIME header of the S/MIME CMS SignedData. When proxy servers receive a message, the proxy server SHOULD inspect the "Content-Target" MIME header.

UAs SHOULD generate a digital signature of whole message body including the "Content-Target" MIME header in order to protect the indication. The proxy server SHOULD validate the signature of whole message body to check the integrity of the indication, even when the "Content-Target" MIME header is not set to whole message body.

This method of indicating the target has less of an impact on proxy

servers that do not support end-to-middle security because these proxy servers do not inspect the MIME header anyway. Also there is less of an impact on UAs that do not support this MIME header, because the UAs will ignore irrelevant MIME headers.

Note: In the last version of this proposal, we used a new parameter in "Content-Disposition" MIME header. As pointed out, the semantic of the header is ambiguous. A new MIME header is better.

Note: There is an alternative option, the use of a new SIP header. This proposed mechanism puts more load on proxy servers to determine the necessity of MIME body handling than using a new SIP header would. However, the proxy can view the indicated MIME body more effectively than using a new SIP header. Also, the validation cost for integrity protection of these headers reduces the merit on using a new SIP header. For the integrity protection of SIP headers, a message body that is application/sipfrag [7] needed. In addition, using a new SIP header could have a negative impact on intermediary proxy servers that do not support end-to-middle security, causing unnecessary processing load. We feel that this MIME header mechanism is not as simple, but it is equally effective.

4. Discovery of the Proxy Server's Policies

A discovery mechanism for proxy server's policies is needed when UAs do not know the policies of the proxy server in a signaling path and the proxy server has its own policy for providing some services. When the proxy server receives a request in which it cannot view some data that must be read in order to proceed or the proxy server receives a request whose sending policy cannot be accepted, the proxy MUST send a response with an error code. If the request is in plain text, the error code SHOULD be 403 (Forbidden) accompanied with a required Content-Type, such as "application/sdp". If the request is in plain text and the digital signature of it is required for an integrity check, the error code SHOULD be 403 (Forbidden) accompanied with a required Content-Type that is "multipart/signed".

Open Issue: How does the error message indicate the Content-Type to be attached with a signature? Can these Content-Type be nested such as "Content-Type: multipart/signed" for "Content-Type: application/sdp"? Is it better to define a new error code for requiring a signature attachment?

If the request contains encrypted data, the error code SHOULD be 493 (Undecipherable), accompanied with a proxy's public key certificate and required Content-Type.

Open Issue: Instead of 493, SHOULD it be 403 that is the same as for requiring a signature attachment? When proxy servers require both of disclosure and the integrity check, how will it be

described?

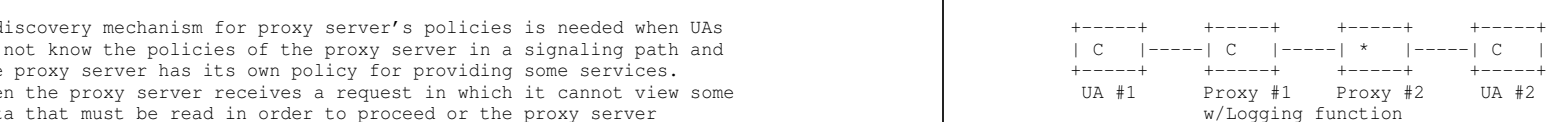
When the UAC receives one of the above error codes, the UAC needs to authenticate the proxy server. Therefore, the error code SHOULD contain the digital signature of the proxy server.

In the worst case, this discovery mechanism requires two messages for each proxy server in the signaling path to establish a session between the UAs. In addition, it requires validation procedures using the digital signatures for all proxy servers. Since this causes a increase in the delay before session establishment, it is recommended that UAs learn in advance the policies of as many proxy servers as they can.

Open Issue: How does this mechanism apply in the case when a proxy server needs to inspect the message body contained in the response? This might be happen when the SDP offer/answer is done with 200/ACK messages.

5. Behavior of UAs and Proxy Servers

We describe here an example of the behavior of UAs and proxy servers in a model in which a proxy server that provides logging services for instant messages exists in a message path as shown in Figure 1.



C: Content that UA #1 allows the entity to inspect
*: Content that UA #1 prevents the entity from inspecting

Figure 1: Deployment example

5.1 UAC Behavior

When a UAC sends an MESSAGE [8] request including encrypted message content for end-to-end and end-to-middle confidentiality, it MUST use S/MIME CMS EnvelopedData to encrypt them. In this example, UA #1 is assumed to know the services and the public key of Proxy #1. UA #1 MUST use CMS EnvelopedData for UA #2 and Proxy #1. UA #1 SHOULD specify Proxy #1 in the "Content-Target" MIME header of the message body to be decrypted.

When the UAC sends a request and needs end-to-end and end-to-middle integrity for the message body, it MUST use S/MIME CMS SignedData to

attach a digital signature. In this example, UA #1 MUST use the CMS SignedData of the contents. UA #1 SHOULD specify Proxy #1 in the "Content-Target" MIME header of the signature to be validated.

When the UAC sends multiple requests to the UAS, the CEK reuse mechanism is beneficial that UAs efficiently encrypt/decrypt data. The CEK reuse mechanism is described in [9][10]. the UAC SHOULD uses the "unprotectedAttrs" field to stipulate reuse of the CEK and indicate its identifier. When the UAC reuses the CEK in the previous request as the KEK, the UAC generate CMS EnvelopedData with the type "KEKRecipientInfo" of "RecipientInfo" attribute.

5.2 UAS Behavior

When a UAS sends a response for the request with this mechanism, using the same type of S/MIME CMS data is recommended. For example, if the UAS receives an INVITE request in which the SDP is encrypted by using CMS EnvelopedData body, the response is RECOMMENDED to be a "200 OK" containing the encrypted SDP based on CMS EnvelopedData body. In the above example, however, the response of the MESSAGE request does not need to use the same type of S/MIME CMS data, since the response does not contain the message content.

In particular, when the CMS EnvelopedData body of the request contains the "unprotectedAttrs" attribute specifying reuse of the CEK, the UAS SHOULD keep the CEK with the identifier specified in the "unprotectedAttrs" attribute.

When the UAS receives a request that uses S/MIME, it decrypts and/or validates the S/MIME bodies as usual.

Even when the UAS receives the request without this mechanism, UAS MAY need end-to-end and end-to-middle confidentiality of the message bodies and/or headers in the response. In this case, the UAS MUST use CMS EnvelopedData to encrypt them. When the UAS sends a response and needs end-to-end and end-to-middle integrity of the message bodies and/or headers, it MUST use CMS SignedData to attach a digital signature. This is the same way the UAC normally performs with this mechanism.

5.3 Proxy Behavior

When a proxy supporting this mechanism receives a message, the proxy server MUST inspect the "Content-Target" MIME header. If the MIME header includes the processing server's own name, the proxy server MUST inspect the specified body.

When the specified body is CMS EnvelopedData, the proxy server MUST

inspect it and try to decrypt the "recipientInfos" field. If the proxy server fails to decrypt that, it SHOULD cancel the subsequent procedure and respond with a 493 (Undecipherable) response if it is a request, or any existing dialog MAY be terminated. If the proxy server succeeds in this decryption, it MUST inspect the "unprotectedAttrs" field of the CMS EnvelopedData. If the attribute gives the key's identifier, the proxy server MUST keep the CEK with its identifier until the lifetime of the CEK is expired. When it receives subsequent messages within the lifetime, it MUST try to decrypt the type "KEKRecipientInfo" of "RecipientInfo" attribute by using this CEK.

When the specified content is CMS SignedData body, the proxy server MUST inspect it and validate the digital signature. If the verification is failed, the proxy server SHOULD reject the subsequent procedure and respond with a 403 (Forbidden) response if the message is a request, or any existing dialog MAY be terminated.

When the proxy server forwards the request, it modifies the routing headers normally. It does not need to modify the S/MIME body.

If a proxy does not support this mechanism and receives a message with the "Content-Target" MIME header, the proxy MUST ignore the header and perform as usual.

6. Content-Target Header Field Use

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [11]. The new header "Content-Target" is defined as a MIME header.

```
Content-Target        =  "Content-Target" HCOLON target-entity
target-entity         =  proxy-uri *(COMMA proxy-uri)
proxy-uri             = ( name-addr / addr-spec )
```

7. Examples

The following examples illustrate the use of the mechanisms defined in the previous sections.

7.1 Request Example for End-to-Middle Confidentiality

In the following example, a UA needs the message content in a MESSAGE request to be confidential and the UA allows a selected proxy server to view the message content. The UA also needs to protect the label of the target content. In addition, the UA needs to reuse the CEK in the subsequent request messages. In the example encrypted message

below, the text with the box of asterisks ("*") is encrypted:

```
MESSAGE alice@atlanta.example.com --> ss1.atlanta.example.com


MESSAGE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 MESSAGE
Date: Fri, 20 June 2003 13:02:03 GMT
Content-Type: multipart/signed;protocol="application/pkcs7-signature";
         micalg=sha1;boundary=boundary1
Content-Length: ...

--boundary1

Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
              name=smime.p7m
Content-Transfer-Encoding: binary
Content-Disposition: attachment;filename=smime.p7m
Content-Target: ss1.atlanta.example.com
Content-Length: ...

*****************************************************************
* (encryptedContentInfo)                                        *
* Content-Type: text/plain                                      *
* Content-Length: ...                                           *
*                                                               *
* Hello.                                                        *
* This is confidential.                                         *
*                                                               *
* (recipientInfos)                                              *
* RecipientInfo[0] for ss1.atlanta.example.com public key       *
* RecipientInfo[1] for bob's public key                         *
*                                                               *
* (unprotectedAttrs)                                            *
* CEKReference                                                  *
*****************************************************************

--boundary1--
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=smime.p7s;handling=required
```



```
[binary data]

--boundary1--
```

7.2 Request Example for End-to-Middle Integrity

In the following example, a UA needs the integrity of the message content in a MESSAGE request to be validated by a selected proxy server. The UA also needs to protect the label of the target content.

```
MESSAGE alice@atlanta.example.com --> ss1.atlanta.example.com


MESSAGE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 MESSAGE
Date: Fri, 20 June 2003 13:02:03 GMT
Content-Type: multipart/signed;protocol="application/pkcs7-signature";
         micalg=sha1;boundary=boundary1
Content-Length: ...

--boundary1

 Content-Type: text/plain
 Content-Length: ...

 Hello.
 This is protected with the signature.

--boundary1--
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: binary
Content-Target: ss1.atlanta.example.com
Content-Disposition: attachment; filename=smime.p7s;handling=required

[binary data]

--boundary1--
```

8. Security Considerations

TBD.

9. IANA Considerations

This document requires a new "Content-Target" MIME header.

10. Acknowledgments

Thanks to Rohan Mahy and Cullen Jennings for their initial support of this concept, and to Jon Peterson for his helpful comments. Jonathan Rosenberg gave us a useful comment.

References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Ono, K. and S. Tachimoto, "Requirements for end-to-middle security in the Session Initiation Protocol (SIP)", draft-ietf-sipping-e2m-sec-reqs-02 (work in progress), May 2004.

[4] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1992.

[5] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[6] Andreasen, F., Baugher, M. and D. Wing, "Session Description Protocol Security Descriptions for Media Streams", draft-ietf-mmusic-sdescriptions-03.txt (work in progress), February 2004.

[7] Sparks, R., "Internet Media Type message/sipfrag", RFC 3420, November 2002.

[8] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[9] Farrell, S. and S. Turner, "Reuse of CMS Content Encryption Keys", RFC 3185, October 2001.

[10] Ono, K. and S. Tachimoto, "Key reuse in S/MIME for SIP", draft-ono-sipping-keyreuse-smime-00 (work in progress), February 2004.

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

Authors' Addresses

Kumiko Ono
Network Service Systems Laboratories
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-shi, Tokyo 180-8585
Japan

EMail: ono.kumiko@lab.ntt.co.jp

Shinya Tachimoto
Network Service Systems Laboratories
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-shi, Tokyo 180-8585
Japan

EMail: tachimoto.shinya@lab.ntt.co.jp

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgment

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

Discussion on Service Providers' Policies with the Session Initiation Protocol (SIP)

draft-ono-sipping-providers-policy-00

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 7, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

Some service providers who operate SIP proxy servers and registrars need to be able to express various types of policies to their customers, such as media policies and security policies. Discussion needs to take place about the types of policies and how they will have an impact on SIP User Agents (UA)s. This document presents an overview of the types of policies that might be available, and how the operations of policies might be executed to aid in advancing the current discussions on session policies.

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 [1].

Table of Contents

1. Overview

Some service providers operate SIP [2] proxy servers and registrars to provide services, such as Voice over IP services and PSTN gateway services. Service providers sometimes place limits on which SIP UAs can connect to its network. However, to allow for interoperability among different SIP UA implementations and the provider's servers, it is necessary to notify SIP UAs of these policies. This document provides examples of a typical service provider's policies, including some that have not yet been discussed in the SIPPING WG. It also discusses possible operations for these policies that will have an impact on the operation of SIP UAs so as to get a better understanding of what session policies need to accomplish.

2. Examples of Service Providers' Policies

We can classify the examples of policies into two categories: media policies and security policies. Session policy work [3] mainly focuses on media policies, and the e2m work [4] mainly focuses on security policies.

- Media Policies
  - Codec restrictions
  - Call admission control for bandwidth management
- Security Policies
  - User authentication for proxy servers
  - Information disclosure for dynamic firewall control
  - Information disclosure for logging services
  - Information disclosure for location-based routing

3. Policy Operations

There are four operations that need to take place for providers' policies to be reflected on UAs, these are listed below.

Operation #1: UAs disclose information that providers utilize to determine session-dependent policies.
Operation #2: Provider's proxy servers instruct the UA about the policies.
Operation #3: UAs comply with or don't comply with the policies.
Operation #4: Providers verify that UAs follow the directives in the policies.

Operation #1 is only required for session-dependent policies. If the policies are statically determined, such as user-by-user basis or for all users, operation #1 is not required. While the media policies have the possibility of being session-dependent and

session-independent, the security policies are always session-independent.

So far in discussions on the mailing list and at the meetings, the WGs have discussed only operations #1 and #2. Operations #3 and #4 are out of scope of the session policies discussion, because only media proxy servers can execute operation #4. However, use cases described in [4] include Operations #3 and #4, because some of these use cases are done in signaling messages, where media proxy servers are not involved. An example is user authentication using HTTP digest authentication in SIP.

4. A Mechanism of Operation #1: UAs disclose information that providers utilize to determine session-dependent policies.

This operation #1 is needed, if the media policies are dependent of session.

There are two mechanism options for this operation, which are both UA driven. Since it is desirable to have the same mechanism to be consistent over the consecutive operations, option#2 which is congruent with the preferred option in operation #2 for the media policies is more desirable.

Option #1: in-band
- Discloses information in messages, such as INVITE/200 or UPDATE/ 200.
- Requires security for end-to-middle, no matter where the information is set; a header or a body.

Option #2: out-of-band
- Discloses information in messages, such as PUBLISH or SUBSCRIBE/NOTIFY.
- Requires the correlation with the session.
- Requires new data definition that contains media attributes of a UAC and the UAS.
- Requires end-to-end security.

5. A Mechanism of Operation #2: Providers instruct UA about the policies

There are two mechanism options: proxy server driven and UA driven mechanisms. Policy servers are assumed to be co-located with proxy servers.

Since option # 1 has several problems, option #2 is generally preferable. The media policies are changeable during a session. The lack of capability of dynamic notification could be a fatal problem

in option #1. Therefore, option #2 is preferable for the media policies.

However, the problems do not come into play for certain use cases. For example, the security policies are not changeable during a session. Some use cases of the security policies are only applied only to a request message, that is to a UAC. Whether to utilize in-band or out-of-band as the preferred mechanism depends on the use cases of the security policies.

Option #1: in-band
* Instructs the policies by proxy server driven.
* Requires a proxy server to return an error response or to add something to a response in order to notify a UAC.
* Requires a proxy server to add something to a request in order to notify the UAS.[OPEN ISSUE]
* Requires middle-to-end security to secure policy information.
* Lacks of a capability of dynamic notification during a session. [OPEN ISSUE]
* Discloses policies to other providers. [OPEN ISSUE]

Option #2: out-of-band
* Instructs the policies by UA driven.
* Requires correlation with the session.
* Requires end-to-end security.

6. A Mechanism of Operation #3: UAs comply with or don't comply with the policies

There are two mechanism options for this operation, which are both UA driven. The media policies feedback on media streams between the UAs. Therefore, for the media policies, this operation, of course, is accomplished with out-of-band.

For the security policies, whether to utilize in-band or out-of-band as a possible mechanism, depends on the use cases. For example, user authentication, logging services, and location-based routing are best done using in-band signaling messages, because information that effect the policies is conveyed within the signaling itself. Dynamic firewall control can be accomplished with either out-of-band or in-band, because information that effect the policies is conveyed separately.

Option #1: in-band
* Appropriate use cases that information that effect the policies is conveyed within the signaling.
* Requires the end-to-middle security.

Option #2: out-of-band
* Appropriate use cases that information that effect the policies is conveyed separately.

7. A Mechanism of Operation #4: Providers verify that UAs follow the directives in the policies.

Media policies need media proxy servers to verify that media streams of the UAs follow the directives. In case of security policies, the proxy servers can reject to transfer the signaling messages unless the UAs follow the directives. This operation is accomplished with in-band.

8. Security Considerations

This document does not introduce a new mechanism.

9. IANA Considerations

This document requires no additional considerations.

10. Acknowledgments

I would like to thank Gonzalo Camarillo, Volker Hilt, Cyrus Shaoul and Shida Schubert.

11 References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Rosenberg, J., "Requirements for Session Policy for the Session Initiation Protocol (SIP)", draft-ietf-sipping-session-policy-req-01 (work in progress), February 2004.

[4] Ono, K. and S. Tachimoto, "Requirements for End-to-middle security in the Session Initiation Protocol(SIP)", draft-ietf-sipping-e2m-sec-reqs-03 (work in progress), July 2004.

Author's Address

Kumiko Ono
Network Service Systems Laboratories
NTT Corporation
9-11, Midori-Cho 3-Chome
Musashino-shi, Tokyo 180-8585
Japan

EMail: ono.kumiko@lab.ntt.co.jp

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 procedures with respect to rights in RFC 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.

SIPPING D. Petrie
Internet-Draft S. Lawrence
Expires: January 7, 2005 Pingtel Corp.
July 9, 2004

A Schema for Session Initiation Protocol User Agent Profile Data Sets
draft-petrie-sipping-profile-datasets-00.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 7, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document defines the requirements and a format for SIP user agent profile data. An overall schema is specified for the definition of profile data sets. The schema also provides for expressing constraints for how multiple sources of profile data are to be combined. This document provides a guide to considerations, policies and syntax for defining data sets to be included in profile data. It also explores some specific data sets to test the requirements, assumptions and syntax.

Table of Contents

1. Motivation

Today all SIP user agent implementers use proprietary means of expressing and delivering user, device, and local network profile information to the user agent. The SIP User Agent Profile Delivery Framework [I-D.ietf-sipping-config-framework] specifies a how SIP user agents locate and retrieve profile data specific to the user, the device, and the local network. It is important for SIP User Agents to be able to obtain and use these multiple sources of profile data in order to support a wide range of applications without undue complexity.

The SIP User Agent Profile Delivery Framework does not define a format for the actual profile data. This document proposes the requirements, a high level schema for, and guide to how these data sets can be defined. The goal is enable any SIP user agent to obtain profile data and be functional in a new environment independent of the implementation or model of user agent. The nature of having profile data from three potential sources requires the definition of policies on how to apply the data in an interoperable way across implementations which may have widely varying capabilities.

The ultimate objective of the framework described in the SIP User Agent Profile Delivery Framework and this document is to provide a start up experience similar to that of users of an analog telephone. From the point of view of a user, you just plug in an analog telephone and it works (assuming that you have made the right arrangements with your local phone company). There is no end user setup required to make an analog phone work, at least in a basic sense. So the objective here is to be able to take a new SIP user agent out of the box, plug it in or install the software and have it get its profiles without human intervention other than security measures. This is necessary for cost effective deployment of large numbers of user agents. All user agents do not provide telephone capabilities, but the use case is applicable to most of the range of user agent capabilities.

2. Introduction

2.1 Requirements Terminology

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in RFC 2119[RFC2119].

2.2 Profile Data Terminology

property - a named configurable characteristic of a user agent. A given property has a well-defined range of possible values. A given property may be defined to have range of values, allow for simultaneous use of many values (as in a list of allowed possibilities), or be a set of related values that collectively form a single profile information item.

setting - the binding of a specific value or set of values to a given property.

profile - a collection of settings to be applied for a specific user, device, or local network.

device - SIP user agent, either software or hardware appliance. This is a logical concept, as there may be no physical dedicated device or it may be part of an assembly of devices. In this document, the terms "user agent" and "device" are interchangeable.

user profile - the profile that applies to a specific user. This is best illustrated by the "hotelling" use case - a user has an association for some period of time with a particular device. The user profile is that set of profile data the user wants to associate with that device (e.g. it rings when someone calls them, it has the users shortcuts installed).

device profile - data profile that applies to a specific device. In the "hotelling" use case, this is the data that is bound to the device itself independent of the user. It relates to specific capabilities of the device and/or preferences of the owner of the device.

local network profile - data that applies to the user agent in the context of the local network. This is best illustrated by roaming applications; a new device appears in the local network (or a device appears in a new network, depending on the point of view). The local network profile includes settings and perhaps policies that allow the user agent to function in the local network.

data set - a collection of properties.

working profile - the set of property values actually set in a SIP User Agent as a result of merging the profiles from all sources; the actual effective profile for the user agent .

merging - the operation of resolving overlapping settings from multiple profiles. Overlap occurs when the same property occurs in multiple profiles (e.g. user, device, local network).

2.3 Overview

In this document requirements are specified for containing and expressing profile data for use on SIP user agents. Though much of this can be considered independent of SIP there is one primary requirement that is not well satisfied through more generic profile data mechanisms. SIP User Agent set up requires the agent to merge

settings, which may overlap, from potentially three different sources; each source must not only be able to provide profile information, but also express policies regarding how the profile settings may be combined with that from other sources.

A schema and syntax is defined in this document to specify properties that may be aggregated to construct profiles. The general design philosophy is that many small data sets provide flexibility to the implementer to support the aggregated set that best matches the capability of the user agent. The actual properties are not defined in this document. However, some examples are explored here to illustrate the proposed mechanisms and to validate the requirements. This document defines a set of considerations, syntax and policies that must be specified when defining data sets. These are to help authors of data set specifications to define data sets that will work in the overall schema defined in this document. The actual specification of these data sets is outside the scope of this document.

3. Requirements

The following section defines some of the requirements that were considered when defining the schema, syntax and policies for generating and applying profile data. This is not an exhaustive list of requirements, but the most significant ones to be satisfied.

3.1 Implementer Extensibility

Implementers must be able to differentiate each implementation. In addition, it does not serve user agent owners and administrators well to require an orchestrated upgrade for all user agent implementations and profile delivery servers before a new capability or feature can be supported with the required profile data. Hence one of the most important requirements is to support the ability of implementers to extend specified standard data sets to include additional related features and flexibility. It MUST be possible to extend a data set without breaking user agents that support that data set. This may require that user agents ignore parts of a data set that it does not implement or extensions that it does support.

3.2 Flexible Capabilities

User agents vary quite widely in their capabilities. Some user agents function like traditional telephones. Some user agents support only text messaging. Some user agents support many media types such as video. Some user agents that function like a telephone have a single line, some have large numbers of lines. There is no

such thing as one size fits all. It MUST be possible for an implementer to choose which data sets to support based upon the capabilities that are supported by the user agent. The schema for containing the profile data MUST support a profile that contains only the data sets that a user agent supports. This allows the profile delivery server to create small profiles for specific devices. However a user agent SHOULD ignore properties for capabilities that it does not support. This allows the profile delivery server to be ignorant of the capabilies of the device. The degree to which the profile delivery server has intelligence of the user agent capabilities is an implementation choice.

3.3 XML

XML is perhaps not really a requirement, but a solution base upon requirements. However it is hard to ignore the desire to utilize readily available tools to manage and manipulate profile data such as XSLT, XPATH and XCAP. The requirement that should be considered when defining the schema and syntax is that many user agents have limited resources for supporting advanced XML operation. The simplest XML construct possible should be used, that support required functionality. Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols [RFC3265] provides useful information in this regard.

3.4 Access Control

Many user agents (e.g. appliances and softphones running on PCs) provide user interfaces that permit the user to edit properties that are logically part of user, device or local network profiles. Operators and administrators would like to be able to specify what an end user can change in those profiles and what an end user is not allowed to change. There may also be sensitive data the user agent requires to function, but that the operator of the system does not want the end user to see. For some properties the system operator may allow the user a fixed set of choices among the supported set of possible values. It MUST be possible to express whether an end user may change a data set property. It MUST be possible to express that a property should not be made visible to the end user. It MUST be possible to express allowable values or ranges that the end user may change a property to. The access control information SHOULD be an optional to the data set. It might be useful if it was possible to express the access control independent of the properties themselves. The access control specification by itself might be useful to express a general policy that the device owner or local network operator wish to impose.

3.5 Data Constraints and Range Definition

There is a need for property value types such as free form text, token/enumerations, integers, real numbers, etc. Many of these properties will have constrained values as opposed to the range of all possible values. These constrains may be due to protocol definitions, implementation limitations, and/or the desire (e.g. by the user, device owner, local network operator) to impose policy on the user agent. The ability to express the property constraints is useful from the perspective of access control as described in the above section. It is also useful to parameterize a user interface (e.g. on the user agent itself or on the profile delivery server) which provides a facility to modify profile data. It MUST be possible for the schema to specify property constraints as ranges or discrete sets of possible values. These constrains SHOULD be optional to the data set. It might be useful if it was possible to express the constraints independent of the properties themselves. The constraints without the property values might be used to specify the capabilities of a particular user agent implementation.

3.6 Support of User, Device, Local Network Sources

[I-D.ietf-sipping-config-framework] specifies a mechanism where the user agent retrieves profile data from as many as three different sources. The separation of the user profile facilitates a hotelling capability and the ability to easily re-assign a user to a different device. The separation of the local network profile facilitates properties specific to operating in the local network in a roaming scenario (e.g. outbound proxy or NAT traversal properties). The local network profile may also impose policy as describe in the next section. The device profile facilitates device capability based properties as well as a means for the device owner to impose policy. The potential sources of profile data add complexity to the user agent that must consolidate these separate profiles into a single working profile. It would be simple if we could define each property as only allowed in one of the profiles. However it overly constrains the profiles and takes away desired functionality. It would also be simpler if we could define one rule for all profile data sets and properties by which we merge the profile (e.g. local network profile overwrites user profile which overwrites device profile for all data). However this too is overly restrictive and eliminates some very useful functionality. The rules to merge profile data sets needs to be defined for each data set. In some cases an entire data set must be considered atomic with a preference as to which profile sources presides over the other. In other cases it makes sense to merge profile data sets, aggregating properties from the data set provided in each of the profiles. It may also be desirable to have

the effect of filtering of data set properties. The desired effect might be for the owner of the device or the local network operator to constrain what values are allowed for properties in the profiles. This may also be the mechanism to facilitate imposing of policy as described in the next section. The operation of resolving overlapping data sets from multiple profiles, regardless of the means or net result, will be referred to as "merging" in this document. A profile MUST have the means to constrain the merging algorithm. [It is not clear whether the merging algorithm can be statically defined by the data set type or if there is a need to specify this as part of the data set (i.e. is this text in a data set definition or must the schema support this expression?). It gives operators and administrators more control if it can be expressed in the schema, but that will lead to more complexity and possible run time problems. Need some more thought and input on this.]

3.7 The Ability to Specify Policy

Local network operators would like to impose policy on users and devices operating in their network. There is a need to constrain the operation and require specific behavior in the network. This might be a simple as to get access to the Internet, user agents must use a specified outbound proxy and NAT traversal mechanism. The network might have limited bandwidth such that the operator would like to constrain codecs or media streams to keep the network functional. The local network may provide emergency service behavior or functionality properties that are more specific than those provided by the device or user profile. The examples here focus on policy from the local network. However the facility to impose policy may be equally useful to the user and device profiles.

It MUST be possible to impose policy in any of the profile sources that constrains, overwrites or modifies properties provided in data sets from other sources.

4. Overall Data Set Schema

This document defines an XML Schema, for SIP Profile Data Sets that provides:

- a base element type from which all settings in other schema definitions inherit (this allows other definitions to specify the content models for ways of combining settings; it is analogous to a C++ virtual base class).
- A set of containers for use when assembling property sets that specify constraints for how settings are to be combined to form a working profile.

- A root element for all property sets (the outermost container).

The full text of the schema is in Appendix A; the following describes the usage of the schema in defining properties and combining them to construct the working profile of a User Agent.

4.1 Data Primitives

Each property in a profile data set is defined using XML Schema Datatypes [W3C.REC-xmlschema-2] and XML Schema Structures [W3C.REC-xmlschema-1]; a property is modelled by an XML element derived from the "setting" element in the SIP Profile Data Set Schema. The element content is the setting value. The XML Schema specifications provide a rich set of mechanisms for defining this data, and XML Namespaces [W3C.REC-xml-names] provide the means to uniquely identify them.

Typically each data set will specify its own namespace. A data set has no structural grouping from an XML perspective. The grouping is logical and identified by its namespace.

4.2 Specifying Access Control

[Specification of access control for settings will be addressed in a future revision of this draft]

4.3 Grouping and Cardinality of Sets of Data

When constructing a property set, the profile delivery server may not be able to know all of the constraints of the User Agent that will receive that property set. In particular, the capabilities of the agent may be limited either intrinsically or by other property sets (some of which may come from other profile sources). The SIP Profile Data Set Schema defines four elements that together express constraints on the valid ways in which the settings within a set can be combined.

4.3.1 property_set

The root element of a property set is "property_set"; it is the container that is provided to the user agent. The elements contained within a property_set form a set of constraints to be "satisfied" by the device; some positive (values to be set), and some negative (prohibited values). An element is "satisfied" iff the working profile of the User Agent matches the constraints of the property_set. The property_set contains all properties that are set from all data sets contained in the profile. The data sets do not have structure other than complex properties which may be defined in

the data set specification. This allows the structured grouping of properties to be based upon the constraints to be applied. The constraints constructs are described in the following sections.

4.3.2 forbid

Each property set contains at most one "forbid" element; settings within the forbid container MUST NOT be in the working profile of the User Agent. This allows one property set to prohibit certain settings in other property sets. For example, a local network property set might forbid the use of high bandwidth codecs, even though the user or device property sets include them.

An empty setting within the forbid element (for example "<foo/>") means that that setting MUST NOT be set to any value.

A non-empty setting within the forbid element (for example "<foo>bar</foo>") MUST NOT be set to the indicated value (or any of the indicated values, in the case of multi-valued settings).

4.3.3 set_all

The "set_all" container element specifies that the User Agent MUST satisfy all of the elements it contains. If the User Agent cannot (due to inherent limitations or conflicting profile constraints) satisfy the elements within a set_all element, then it MUST NOT use any of them, and the set_all profile element is considered not to have been satisfied.

4.3.4 set_one

The "set_one" container element is an ordered list of elements; it specifies that the User Agent MUST satisfy the first of the contained elements that it can without conflicting with other constraints. If the User Agent cannot (due to inherent limitations or conflicting profile settings) satisfy one of the contained settings, then the set_one profile element is considered not to have been satisfied.

4.3.5 set_any

The "set_any" container element specifies optional settings; the User Agent SHOULD include any of the contained elements in its working profile, unless inherent limitations or other profile settings conflicts with them. A set_any element is always satisfied, even if none of the elements it contains are satisfied.

4.4 Common Types

[The schema will also define a set of common types that are used in defining data sets (e.g. name-addr) in a future version of this draft.]

4.5 Merging Property Sets

[Some discussion is needed here on conflict resolution. Reviewers are encouraged to consider the implications of conflicting property sets, especially when different property sets are provided to the same device possibly from different sources.]

5. Defining Data Sets

This section defines considerations and information that must be defined when specifying a new data sets. This is intended to be a guide to authors writing specifications defining new data sets or extensions to existing ones.

5.1 Data Set Properties Definitions

Data set specification documents should contain a section which defines the meaning of all of the properties contained in the data set. The objective is to define the property such that implementers have a clear definition and semantics to interpret properties in a consistent way. User agents not only need to use the same profile content, they need to apply the properties in a consistent way to achieve true interoperability.

The following information should be defined for each property in a data set:

description - Describe the meaning and application of the property.

cardinality - Define how many of this property may occur in a data set (e.g. zero, one or many) as well as its relationship to any other properties in this or other data sets.

default value - Define the default value of this property if it is not set. Describe if the default is different if the property is present and not set vs. completely absent from the data set. Define if the default varies in relation to another property.

5.2 Data Set Schema Definition

A data set should define a new XML namespace [W3C.REC-xml-names] to scope all of the properties that are defined in the name space. properties may be simple (i.e. having a single value) or they may be complex (i.e. a container or structure of values). Each property in the data set SHOULD inherit from the "setting" element. Complex

properties and all of their child elements each should inherit from "settings" as well.

5.3 Merging Different Sources of a Data Set

Collisions may occur on a data set if multiple sources (e.g. user, device and/or local network) provide properties for that data set. Data set specifications MUST define the policy and algorithm by which to resolve the conflict. This resolution of conflict from multiple sources is called merging. The data set specification can determine how merging occurs for that data set. The author may choose to combine, apply a policy of mutually exclusive ordered preference (i.e. the entire atomic data set is used from one profile source in a defined order of preference), or well defined combination of these or other algorithms.

[Should we define some common algorithms here that authors can refer to? Perhaps the schema should allow this to be expressed as part of the data set?]

6. Candidate Data Sets

The following sections name some of the candidate data sets that might be defined. These data sets can be aggregated to form profiles appropriate to the capabilities of a user agent implementation.

6.1 SIP Protocol Data Set

The lowest common denominator set of properties common to all SIP user agents of any capability.

6.2 Media Data Set

Codecs and media streams

6.3 Identity Data Set

AORs and lines

6.4 HTTP Protocol Data Set

Server settings. Proxy for clients.

6.5 STUN Protocol Data Set

6.6 TURN Protocol Data Set

6.7 Address Book

6.8 Buddy List

6.9 SIP Digit Maps Data Set

7. Example Data Set Definitions

To test the schema a few example data sets are defined here.

[The examples in this section are contained in this document for convenience. At some point in this document's lifecycle they will be split out as separate drafts.]

7.1 SIP Protocol Data Set

The SIP Protocol Data Set is intended the be the lowest common denominator among all user agent types regardless of capability. This data set contains properties that all user agents require. That does not mean that all of these properties are mandatory.

7.1.1 Data Set Properties Definitions

transport_protocol - This property contains properties related to a SIP transport protocol. It names the transport protocol, defines whether the protocol is enabled or not and defines the port to which that protocol is bound. If the protocol is named it defaults to enabled if not explicitly set. If the port property is not set, it defaults to the default specified by the specification which binds the protocol to SIP. The user agent should enable all the set transport protocols that are supported by the user agent. The user agent ignores protocol bindings that it does not support. The user agent may default transport protocols to enabled, that it supports, if a protocol property for that transport protocol is not present in the data set.

outbound-proxy - The default outbound proxy, through which all SIP requests, not explicitly routed, should be sent. The format of this parameter is of name-addr as specified in [RFC3261]. This property is optional. If absent or not set, SIP requests are sent to directly to the URI of the request. If set the effect of this property is to add a loose route as defined in [RFC3261] for the next hop destination.

The following is an example instance of the SIP protocol data set.

```
   <property_set>
      <forbid>
         <transport_protocol>
            <name>UDP</name>
            <port>5060</port>
         </transport_protocol>
      </forbid>
      <set_any>
         <transport_protocol>
            <name>TCP</name>
            <port>5060</port>
         </transport_protocol>
         <transport_protocol>
            <name>TLS</name>
            <port>5061</port>
         </transport_protocol>
      </set_any>
      <set_all>
         <outbound_proxy>sip:outproxy.example.com</outbound_proxy>
      </set_all>
   </property_set>
```

7.1.2 Data Set Schema Definition

The following is the schema for the SIP protocol data set.

```
   <?xml version='1.0' encoding='iso-8859-1' standalone='yes'?>
   <!--
       XML Schema for SIP Protocol core Data Sets
     -->
   <schema
   xmlns:spds='http://sipfoundry.org/schema/profile-data-sets-00'
   targetNamespace='http://sipfoundry.org/schema/profile-data-sets-00'
   xmlns='http://www.w3.org/2001/XMLSchema'
       >
    <annotation>
      <documentation>
        SIP Protocol Properties.
      </documentation>
    </annotation>
    <element name="transport_protocol" group="spds::setting">
     <annotation>
      <documentation>
       Container for the properties for a single transport protocol
       binding for SIP.
      </documentation>
```



```
     </annotation>
      <complexType>
       <sequence>
        <element ref="spds:name" />
        <element ref="spds:port" />
       </sequence>
      </complexType>
    </element>
    <element name="name" group="spds::setting">
     <annotation>
      <documentation>
       Name of the specific transport protocol
      </documentation>
     </annotation>
     <simpleType type="spds:transport"/>
    </element>
    <element name="port" group="spds::setting">
     <annotation>
      <documentation>
       Port binding for the transport protocol
      </documentation>
     </annotation>
     <simpleType type="spds:port_num"/>
    </element>
    <element name="outbound_proxy" group="spds::setting">
     <annotation>
      <documentation>
       The next hop proxy for SIP requests without a defined
       route set.  Value is of name-addr format.  There should
       probably be a type defined for name-addr that outbound_proxy
       inherits from.
      </documentation>
     </annotation>
     <simpleType />
    </element>
   </schema>
```

7.1.3 Merging Different Sources of a Data Set

The entire SIP Protocol Data Set is considered atomic when merging from multiple data set. The entire data set is used from the first of the following sources that provides the data set: local network, device or user profile.

7.2 Media Data Set

The following is example data that should be defined in the media data set:

Video
- codec1
- codec 2

Audio
- G.711
- G.722.1
- G.729A
- ILBC

Text
- IM
- realtime-text

maximum number of streams/session
maximum number of streams total
maximum allowed bandwidth per stream
IP addresses/ports
TOS marking

8. Example Use Cases

8.1 Merge Two Data Sets

(personal and local service speed dial lists)

8.2 Policy Filtering

(allowed and disallowed codecs)

8.3 Override

(device prefers default ports 5060, local net requires port 11000)

9. Security Considerations

Security is mostly a delivery problem. The delivery framework SHOULD provide a secure means of delivering the profile data as it may contain sensitive data that would be undesirable if it were stolen or sniffed. Storage of the profile on the profile delivery server and user agent is an implementation problem. The profile delivery server and the user agent SHOULD provide protection that prevents unauthorized access of the profile data. The profile delivery server and the user agent SHOULD enforce the access control policies defined

in the profile data sets if present.

[The point of the access control construct on the data set is to provide some security policy on the visibility and ability to change sensative properties. Does the access control mechanism also create a security problem where the local network can set or hide properties from the user?]

Some transport mechanisms for delivery of the profile data do not provide a secure means of delivery. In addition some user agents may not have the resources to support the secure mechanism used for delivery (e.g. TLS).

[Should we specify a mechanism to symmetrically encrypt the profile (e.g. AES) and a key format? The profile delivery server would encrypt the profile before delivery and the user agent would decrypt the profile after collecting the appropriate credential information to generate the correct key. Many user agents support a mechanism like this to overcome insecure profile delivery mechanisms. It is lighter weight foot print wise and to implement than adding TLS.]

10 References

[I-D.ietf-sipping-config-framework] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-03 (work in progress), May 2004.

[I-D.sinnreich-sipdev-req] Butcher, I., Lass, S., Petrie, D., Sinnreich, H. and C. Stredicke, "SIP Telephony Device Requirements, Configuration and Data", draft-sinnreich-sipdev-req-03 (work in progress), February 2004.

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[W3C.REC-xml-names]

Bray, T., Hollander, D. and A. Layman, "Namespaces in XML", W3C REC-xml-names, January 1999, <http://www.w3.org/TR/REC-xml-names>.

[W3C.REC-xmlschema-1]
Thompson, H., Beech, D., Maloney, M. and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC-xmlschema-1, May 2001, <http://www.w3.org/TR/xmlschema-1/>.

[W3C.REC-xmlschema-2]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes", W3C REC-xmlschema-2, May 2001, <http://www.w3.org/TR/xmlschema-2/>.

Authors' Addresses

Daniel Petrie
Pingtel Corp.
400 W. Cummings Park
Suite 2200
Woburn, MA 01801
US
Phone: "Dan Petrie (+1 781 938 5306)" <sip:dpetrie@pingtel.com>
EMail: dpetrie@pingtel.com
URI: http://www.pingtel.com/

Scott Lawrence
Pingtel Corp.
400 W. Cummings Park
Suite 2200
Woburn, MA 01801
US
Phone: "Scott Lawrence (+1 781 938 5306)" <sip:slawrence@pingtel.com>
EMail: slawrence@pingtel.com
URI: http://skrb.org/scott/

Appendix A. SIP UA Profile Schema

```
<?xml version='1.0' encoding='iso-8859-1' standalone='yes'?>
<!DOCTYPE schema [
<!ENTITY % doc_src
"http://scm.sipfoundry.org/rep/ietf-draft/petrie/profile-data-sets">
]>
<!--
```



```
      XML Schema for SIP Profile Data Sets
    -->
  <schema
  xmlns:spds='http://sipfoundry.org/schema/profile-data-sets-00'
  targetNamespace='http://sipfoundry.org/schema/profile-data-sets-00'
  xmlns='http://www.w3.org/2001/XMLSchema'
    >
   <annotation>
     <documentation>
       Proposed XML metalanguage for the description of
       SIP User Agent Profile Data Sets.
     </documentation>
     <documentation source='%doc_src;'/>
   </annotation>
  <!-- Types
    Later versions of the Internet-Draft of which this is a part may
    include additional data type definitions and entities useful
    in defining SIP data.
   -->
   <simpleType name="port_num">
    <restriction base="integer">
     <minExclusive>0</minExclusive>
     <maxInclusive>65535</maxInclusive>
    </restriction>
   </simpleType>
   <simpleType name="transport_protocol">
     <restriction base="string">
       <enumeration value="TCP"/>
       <enumeration value="UDP"/>
       <enumeration value="TLS"/>
     </restriction>
   </simpleType>
  <!-- Elements
    Later versions of the Internet-Draft of which this is a part may
    include additional data type definitions and entities useful
    in defining SIP data.
   -->
   <element name="property_set">
     <annotation>
       <documentation>
       The property_set element is the root element returned in
       response to a request for a profile data set.
       </documentation>
```

```
      </annotation>
      <complexType>
       <sequence>
        <element ref="spds:forbid" minOccurs="0" maxOccurs="1"/>
        <sequence minOccurs="0" maxOccurs="unbounded">
          <choice>
            <element ref="spds:set_any" />
            <element ref="spds:set_all" />
          </choice>
        </sequence>
      </sequence>
     </complexType>
    </element>
    <element name="setting" type="anyType" abstract="true">
     <annotation>
      <documentation>
       The 'setting' element is an abstract used as the basis for the
       definition of the setting elements in property schemas derived
       from this one.
       It serves here as a placeholder in constructing the content
       models for the container elements used to group settings into
       sets.
      </documentation>
      <documentation source='%doc_src;'/>
     </annotation>
    </element>
    <element name="forbid">
     <complexType>
      <sequence minOccurs="1" maxOccurs="unbounded">
       <element ref="spds:setting"/>
      </sequence>
     </complexType>
    </element>
    <element name='set_any'>
     <annotation>
      <documentation>
       Contains some number of settings; the user agent MAY include
       none, any, or all of the contained settings, except those also
       listed in a 'forbid' element of the current configuration.
      </documentation>
     </annotation>
     <complexType>
      <sequence minOccurs="1" maxOccurs="unbounded">
       <choice>
```



```
        <element ref="spds:setting" />
        <element ref="spds:set_all" />
        <element ref="spds:set_one" />
       </choice>
      </sequence>
     </complexType>
    </element>
    <element name='set_all'>
     <annotation>
      <documentation>
       Contains some number of settings; the user agent MUST
       include all of the contained settings.
      </documentation>
     </annotation>
     <complexType>
      <sequence minOccurs="1" maxOccurs="unbounded">
       <choice>
        <element ref="spds:setting" />
        <element ref="spds:set_any" />
        <element ref="spds:set_one" />
       </choice>
      </sequence>
     </complexType>
    </element>
    <element name='set_one'>
     <annotation>
      <documentation>
       Contains an ordered sequence of settings;
       the user agent MUST include the first of the contained
       settings of which is capable and which is not listed
       in a 'forbid' element of the working profile,
      </documentation>
     </annotation>
      <complexType>
       <sequence minOccurs="1" maxOccurs="unbounded">
        <choice>
         <element ref="spds:setting" />
         <element ref="spds:set_any" />
         <element ref="spds:set_all" />
        </choice>
       </sequence>
      </complexType>
    </element>
   </schema>
```

Appendix B. Acknowledgments

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgment

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

SIPPING
Internet-Draft
Expires: January 6, 2005

J. Rosenberg
dynamicsoft
G. Camarillo
Ericsson
D. Willis
dynamicsoft
July 8, 2004

A Framework for Consent-Based Communications in the Session Initiation Protocol (SIP)

draft-rosenberg-sipping-consent-framework-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 6, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

The Session Initiation Protocol (SIP) supports communications across many media types, including real-time audio, video, text, instant messaging, and presence. In its current form, it allows session invitations, instant messages, and other requests to be delivered from one party to another without requiring explicit consent of the recipient. Without such consent, it is possible for SIP to be used

for malicious purposes, including spam and denial-of-service attacks. This document identifies a framework for consent-based communications in SIP.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1] supports communications across many media types, including real-time audio, video, text, instant messaging and presence. This communication is established by the transmission of various SIP requests (such as INVITE and MESSAGE [2]) from an initiator to the recipient, with whom communication is desired. Although a recipient of such a SIP request can reject the request, and therefore decline the session, a SIP network will deliver a SIP request to the recipient without their explicit consent.

Receipt of these requests without explicit consent can cause a number of problems in SIP networks. These include spam and DoS (Denial of Service) attacks. These problems are described in more detail in a companion requirements document [draft-rosenberg-sipping-consent-reqs-00.txt].

This specification defines a basic framework for adding consent-based communication to SIP.

2. Relays

A central concept in this framework is that of a relay. A relay is defined as any SIP server, be it a proxy, back-to-back user agent or some hybrid, which receives a request and translates the request URI into one or more next hop URIs to which it then delivers a request. So, an essential aspect of a relay is that of translation.

When a relay receives a request, it translates the request URI into one or more additional URIs. Or, more generally, it can create outgoing requests to one or more additional URIs. The translation operation is what creates the consent problem. Since the translation operation can result in more than one URI, it is the source of amplification. Servers that do not perform translations, such as outbound proxy servers, do not cause amplification.

Since the translation operation is based on local policy or local data (such as registrations), it is the vehicle by which a request is delivered directly to an endpoint, when it would not otherwise be possible to. In other words, if a spammer has the address of a user, sip:user@example.com, it cannot deliver a MESSAGE request to the user agent of that user without having access to the registration data that maps sip:user@example.com to the UA on which that user is present. Thus, it is the usage of this registration data, and more generally, the translation logic, which must be authorized, in order to prevent undesired communications.

3. Reference Architecture

The reference architecture is shown in Figure 1. In this architecture, a UAC wishes to send a message to a request URI representing a resource in domain A (sip:resource@A). This request may pass through a local outbound proxy (not shown), but eventually arrives at a server authoritative for domain A. This server, which acts as a relay, performs a translation operation, translating the request URI into one or more next hop URIs, which may or may not belong to domain A. This relay may be a proxy server of a URI-list service, for instance.

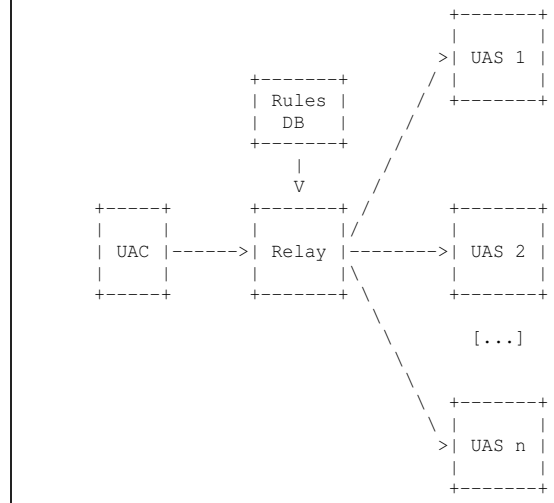


Figure 1

4. Structure of a Permission

This framework centers on the idea that a relay will only perform a translation if a permission is in place authorizing that translation. As such, the notion of a permission is another key part of this framework. A permission is an object, represented in XML, that contains several pieces of data:

Identity of the Sender: A URI representing the identity of the sender for whom permissions are granted.

Identity of the Recipient: A URI representing the target of the translation. The permission grants ability for the sender to send requests, and for a relay receiving those requests to forward them to this URI. This is also called the recipient URI.

Operations Permitted: A set of specific methods or qualifiers for which the permission applies. For example, the permission may only grant relaying for INVITE or MESSAGE, or for MESSAGE with specific MIME types.

Signature: A digital signature over the rest of the permission, signed by an entity that can identify itself as the recipient URI. The signature is not always present.

Permissions are installed on a resource by resource basis. That is, for each target URI to which a request is sent, there is a set of permissions installed for that URI. Each permission has the content described above.

It is important to note that the permission itself does not depend on, or contain, the identity of a target URI (i.e., the input). As such, if a request is sent to sip:resource1@A and to sip:resource2@A, and for both targets, the same permission was installed, allowing requests from the sender to be relayed to sip:resource@B, that same permission would allow the translation to take place for both targets.

A natural format for representing permissions appears to be the common policy format [4]. This format is also used for presence permissions.

5. Attempting Communication

When a UA sends a request to a target resource, the request eventually arrives at a server that is authoritative for the domain in the request URI. The server may require, as part of its processing logic, the relaying of the request to one or more next hops. If such relaying is required, the server first authenticates the sender of the request. Such authentication can be done using the SIP identity mechansim [5]. Once the sender is authenticated, the server checks its permission database for that target resource. It looks for permissions containing senders whose URI matches the identity of the sender of the request. Of those that are found, the server checks to see if the permitted translated URI matches the URIs to which the server wishes to relay the request.

If at least one of the next hops to which the server wishes to relay have not been permitted, the server rejects the request with a 470 (Consent Needed) response. The 470 response code indicates that the request couldn't be relayed because at least one permission was not present. The error response can contain a body, which contains a list of URIs for the translations for which permissions have not yet been obtained. This is effectively an instruction for the sender to go off, and obtain permissions from those URIs.

6. Requesting a Permission

If the attempt to communicate was rejected with a 470 (Consent Needed) response, the client knows that it must obtain some number of permissions in order for the communications to take place. The error response will include a list of URIs for which permission must be obtained. To obtain permission, the client sends a CONSENT request to each of the URIs it learned from the body of the error response. These URIs typically route to the relay, which will forward them on to the destinations whose permissions have not been obtained yet. The CONSENT request carries a Consent-Methods header field which indicates for which methods consent is being requested.

When the CONSENT request arrives at the relay, the relay adds a Permission header field which contains a URI that the receiver can use to upload a permission (e.g., the receiver can use XCAP to upload an XML-based permission document). Then, the relay forwards the request towards its destination.

If there are several relays between the sender and the final destination, those CONSENT requests may also fail if permissions have not yet been obtained, in which case the process recurses. Eventually, the client will have sent a request to all of the relays at the leaves of the translation tree between the sender and the final destinations.

7. Waiting for Permissions

A CONSENT request is responded with a 202 (Accepted) response, which carries a URI in a Call-Info header field (wait-permission purpose) where the client can SUBSCRIBE to using the wait-permission event package. This event package models the state of the permission granted to the client for communicating with the target URI. When a permission is granted, the state changes, and the client receives a NOTIFY. This NOTIFY contains the permission(s) that have been granted for the sender.

Usage of an event package has the benefit that the client can come back at any time and do a query SUBSCRIBE to see if permissions were

granted, or it can wait for them to be granted, and find out when. There is no requirement that the client use this event package to wait. For some requests, it may not be important for the sender to find out when permission is granted (e.g., a presence subscription).

8. Granting a Permission

On reception of a CONSENT request, if the user wishes to grant a permission, XCAP is used, just as it is today in presence. The owner of the target resource would use contact the URI in the Permission header field of the CONSENT request and use XCAP to place the permission into a document containing the list of permissions for that target resource.

The XCAP server needs to make sure that the entity uploading the permission document is the same as the destination of the CONSENT request. This is done by inserting a URI in the Permission header field of the CONSENT request which is long and random enough so that it cannot be guessed. In addition, the CONSENT request is delivered to the user using a SIPS URI. Then, the server inserting such a URI relies on the SIP routing infrastructure to deliver the CONSENT request to its proper destination.

> If the SIP routing infrastructure is compromised, it could route the CONSENT request to an attacker so that the attacker could authorize requests addressed to a victim. Nevertheless, if the SIP routing infrastructure gets compromised, many types of attacks much worse than this are possible. So, relaying on the SIP routing infrastructure seems like a sensible choice.

Using XCAP to grant permissions will require the definition of a new application usage. We note that this usage appears to be a generalization of the presence rules usage currently defined [PRES-RULES].

8.1 Permission Servers

We have just described how a user agent that receives a CONSENT request can use XCAP to grant certain permissions. Nevertheless, users are not on-line all the time and, so, sometimes are not able to receive CONSENT requests.

This issue is also found in presence, where a user's status is reported by a presence server instead of by the user's user agents, which can go on and off-line. Similarly, we define permission servers. Permission servers are network elements that act as SIP UAs and handle CONSENT requests for a user.

Permission servers inform users about new CONSENT requests using the "grant-permission" event package. The user associated with the target URI SUBSCRIBEs to the "grant-permission" event package at the permission server. This event package models the state of all pending CONSENT requests for a particular resource, for which permissions do not yet exist. When a new CONSENT request arrives for which permissions have not been granted, a NOTIFY is sent to the user. This informs them that permission is needed for a particular sender. The NOTIFY contains information on the operation which was requested.

> There is a strong similarity between the watcherinfo event package and the grant-permission event package. Indeed, the grant-permission package is effectively a superset of watcherinfo. Once in place, presentities could use the grant-permission event package for presence in addition to all other services for which opt-in is being provided.

9. Retrying the Original Request

The sender learns about permissions through the wait-permission event package. Once it has obtained permissions for all of the resources that were identified in the 470 (Consent Needed) response, the client can retry the original request.

10. Permission Revocation

At any time, if a client wants to revoke any permission, it uses the XCAP URI that received in the CONSENT message or through the grant-permission event package. If a client lost this URI for some reason, it would need to wait until it received a new request and respond with a 470 (Consent Needed) response. The client would get the URI in a new CONSENT request.

OPEN ISSUE: if we defined the Permission header field so that it can be present in any request, and not only in CONSENT requests, the relay could add this header field to every request directed to the user which used SIPS.

11. Use Cases

The following use cases exhibit how the framework works.

11.1 Basic Flow with No Permission Server

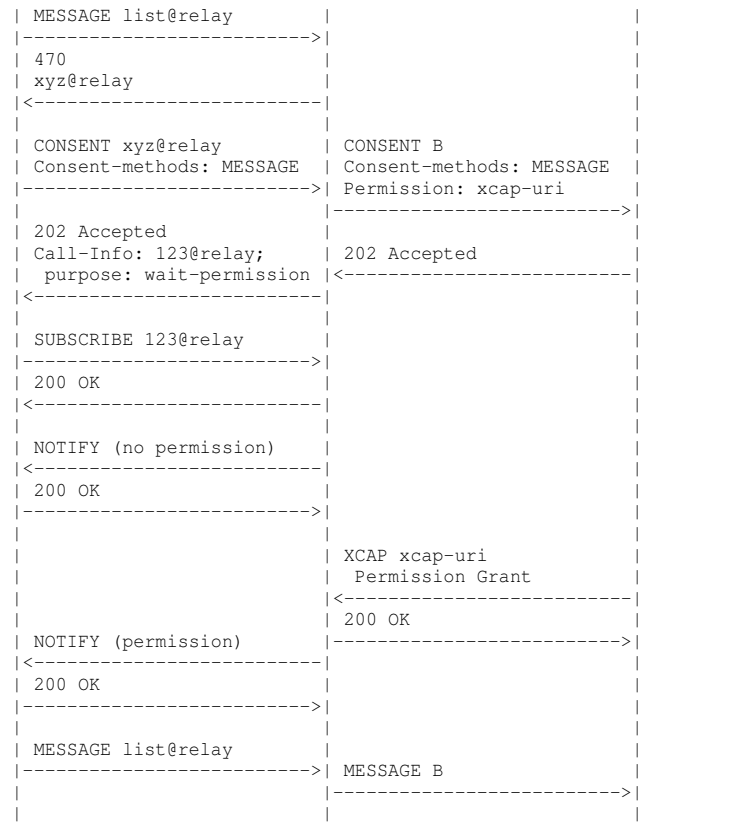
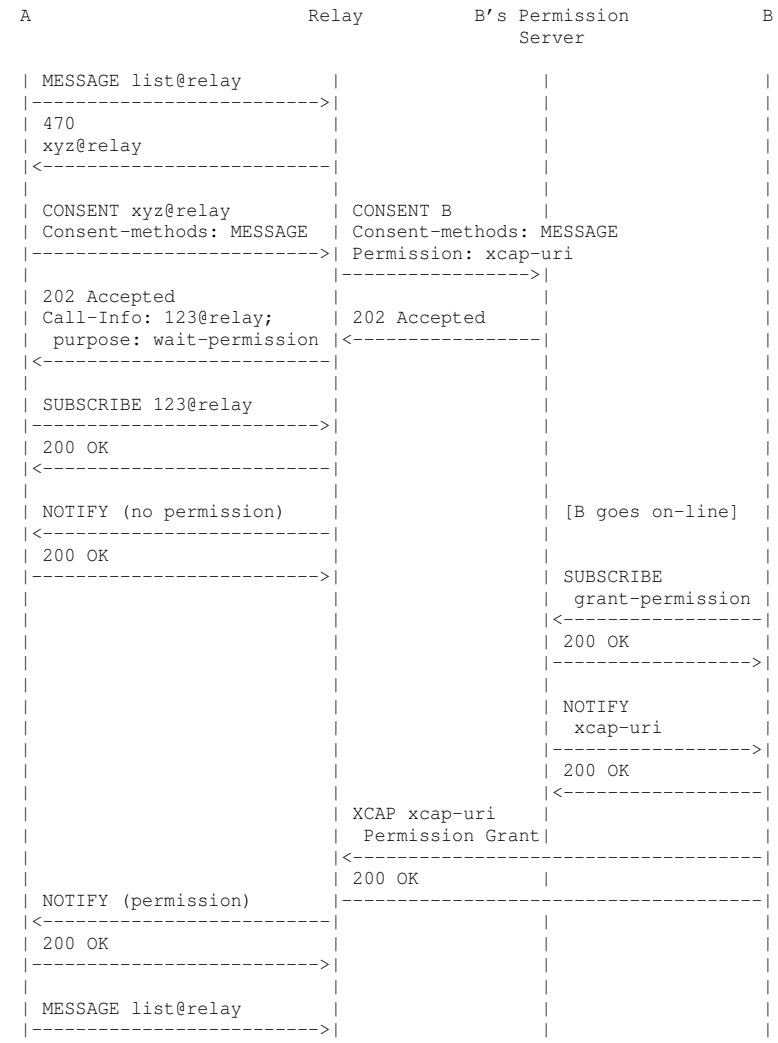


Figure 2

Alternatively, the Call-Info header field could have been inserted by B directly. In this case, A would SUBSCRIBE to B, instead of subscribing to the Relay.

11.2 Basic Flow with a Permission Server



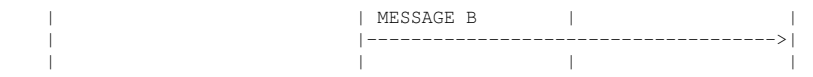


Figure 3

12. References

12.1 Normative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

12.2 Informative References

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

[4] Schulzrinne, H., "Common Policy", draft-ietf-geopriv-common-policy-00 (work in progress), February 2004.

[5] Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-02 (work in progress), July 2003.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Dean Willis
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75028
USA

EMail: dean.willis@softarmor.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.

SIP
Internet-Draft
Expires: January 6, 2005

J. Rosenberg
dynamicsoft
G. Camarillo
Ericsson
D. Willis
dynamicsoft
July 8, 2004

Requirements for Consent-Based Communications in the Session Initiation Protocol (SIP)

draft-rosenberg-sipping-consent-reqs-00.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 6, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

The Session Initiation Protocol (SIP) supports communications across many media types, including real-time audio, video, text, instant messaging, and presence. In its current form, it allows session invitations, instant messages, and other requests to be delivered from one party to another without requiring explicit consent of the recipient. Without such consent, it is possible for SIP to be used

for malicious purposes, including spam and denial-of-service attacks. This document identifies a set of requirements for extensions to SIP that add consent-based communications.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1] supports communications across many media types, including real-time audio, video, text, instant messaging, and presence. This communication is established by the transmission of various SIP requests (such as INVITE and MESSAGE [4]) from an initiator to the recipient, with whom communication is desired. Although a recipient of such a SIP request can reject the request, and therefore decline the session, a SIP network will deliver a SIP request to the recipient without their explicit consent.

Receipt of these requests without explicit consent can cause a number of problems in SIP networks. These include spam and DoS (Denial of Service) attacks. These problems have plagued email. Fortunately, most SIP networks, at time of writing, were not interconnected with each other, and so the incidences of such problems have been lower. However, once such broad interconnection occurs, these problems will arise. Therefore, it is important to address them proactively, before it is too late.

This document elaborates on the problems posed by the current open model in which SIP was designed, and then goes on to define a set of requirements for adding a consent framework to SIP.

2. Problem Statement

In SIP networks designed according to the principles of RFC 3261 [1] and RFC 3263 [2], anyone on the Internet can create and send a SIP request to any other SIP user, by identifying that user with a SIP URI. The SIP network will usually deliver this request to the user identified by that URI. It is possible, of course, for network services, such as call screening, to block such messaging from occuring, but this is not widespread and certainly not a systematic solution to the problem under consideration here.

Once the SIP request is received by the recipient, the user agent typically takes some kind of automated action to alert the user about receipt of the message. For INVITE requests, this usually involves "ringing the phone", or creating a screen pop. These indicators frequently convey the subject of the call and the identity of the caller. Due to the real-time nature of the session, these alerts are typically disruptive in nature, so as to get the attention of the user.

For MESSAGE requests, the content of the message is usually rendered to the user.

SUBSCRIBE [3] requests do not normally get delivered to the user agents residing on a user's devices. Rather, they are normally processed by network-based state agents. The watcher information event package allows a user to find out that such requests were generated for them, affording the user the opportunity to approve or deny the request. As a result, SUBSCRIBE processing, and most notably presence, already has a consent-based operation. Nevertheless, this already-existing consent mechanism for SIP subscriptions does not protect network agents against DoS attacks.

There are two principal problems that arise when MESSAGE and INVITE requests can be delivered to user agents directly, without their consent. The first is spam. For INVITE requests, this takes the form of typical "telemarketer" calls. A user might receive a stream of never-ending requests for communications, each of them disrupting the user and demanding their attention. For MESSAGE requests, the problem is even more severe. The user might receive a never-ending stream of screen pops that deliver unwanted, malicious, or otherwise undesired content.

The second problem is DoS attacks. SIP proxies provide a convenient relay point for targeting a message to a particular user or IP address, and in particular, relaying to a recipient which is often not directly reachable without usage of the proxy. Worse, some proxies or back to back user agents generate multiple outgoing requests upon receipt of an incoming request. This occurs in forking proxies, and in URI-list services. Examples of URI-list services are subscriptions to resource lists, dial out conference servers, and MESSAGE URI-list services. These SIP elements can be used as an amplifier, allowing the transmission of a single SIP request to flood packets to a single recipient or network. For example, a user can create a buddy list with 100 entries, each of which is a URI of the form "sip:identifier@target-IP", where target-IP is the IP address to which the attack is to be directed. Sending a single SIP SUBSCRIBE request to such a list will cause the resource list server to generate 100 SUBSCRIBE requests, each to the IP address of the target, which does not even need to be a SIP node.

> Note that the target-IP does not need to be the same in all the URIs in order to attack a single machine. For example, the target-IP addresses may all belong to the same subnetwork, in which case the target of the attack would be the access router of the subnetwork.

Though the spam and DoS problems are not quite the same, both can be alleviated by adding a consent-based communications framework to SIP. Such a framework keeps servers from relaying messages to users without their consent.

The framework for SIP URI-list services [draft-ietf-sipping-uri-services-00.txt] identifies these two problems (spam and DoS attacks) in the context of URI-list services. That framework mandates the use of opt-in lists, which are a form of consent-based communications. The reader can find an analysis on how a consent-based framework help alleviating spam-related problems in [draft-rosenberg-sipping-spam-00.txt]

3. Requirements

The following identify requirements for a solution that provides consent-based communications in SIP.

REQ 1: The solution must keep relays from delivering a SIP message to a recipient unless the recipient has explicitly granted permission for receipt of that type of message.

REQ 2: The solution shall prevent SIP servers from generating more than one outbound request in response to an inbound request, unless permission to do so has been granted by the resource to whom the outbound request was to be targeted.

REQ 3: The permissions shall be capable of specifying that messages from a specific user, identified by a SIP AoR, are permitted.

REQ 4: It shall be possible for a user with a particular AoR to specify permissions separately for each resource that wishes to relay requests to that AOR.

REQ 5: The permissions shall be capable of specifying that only certain types of messages, such as INVITE or MESSAGE request, are permitted from a user.

REQ 6: It shall be possible for a user to revoke permissions at any time.

REQ 7: It shall be possible for the users to specify that permissions are time limited, and must be refreshed after expiration.

REQ 8: It shall not be required for a user or user agent to store information in order to be able to revoke permissions that were previously granted for a relay resource.

REQ 9: The solution shall work in an inter-domain context, without requiring pre-established relationships between domains.

REQ 10: The solution shall work for all current and future SIP methods.

REQ 11: The solution shall be applicable to forking proxies.

REQ 12: The solution shall be applicable to URI-list services, such as resource list servers, MESSAGE URI-list services, and conference servers performing dial-out functions.

REQ 13: The solution shall be applicable to both stored and request-contained URI-list services.

REQ 14: The solution shall allow anonymous communications, as long as the recipient is willing to accept anonymous communications.

REQ 15: If the recipient of requests wishes to be anonymous, it shall be possible for them to grant permissions without a sender knowing their identity.

REQ 16: The solution shall prevent against attacks that seek to undermine the underlying goal of consent. That is, it should not be possible to "fool" the system into delivering a request for which permission was not, in fact, granted.

REQ 17: The solution shall not require the recipient of the communications to be connected to the network at the time communications is attempted.

REQ 18: The solution shall note require the sender of a communications to be connected at the time that a recipient provides permission.

REQ 19: The solution should not, in and of itself, create substantial additional messaging. Doing so defeats some of the purpose of the solution.

REQ 20: The solution should scale to Internet-wide deployment.

4. Security Considerations

Security has been discussed throughout this specification.

5 Informative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

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

[4] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Dean Willis
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75028
USA

EMail: dean.willis@softarmor.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.

SIPPING
Internet-Draft
Expires: January 17, 2005

J. Rosenberg
dynamicsoft
G. Camarillo
Ericsson
July 19, 2004

Examples of Network Address Translation (NAT) and Firewall Traversal for the Session Initiation Protocol (SIP) draft-rosenberg-sipping-nat-scenarios-03

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 17, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document contains a set of examples about how to establish sessions through Network Address Translators (NATs) using the Session Initiation Protocol (SIP). NAT traversal for SIP is accomplished using Interactive Connectivity Establishment (ICE), which allows the media streams to work, in addition to the SIP extension for symmetric response routing, which allows SIP itself to flow through NAT. The examples cover a range of network topologies and use cases. This

variability helps to demonstrate that the ICE methodology always works, and that a common client algorithm, independent of the network topology and deployment configuration, results in the best connectivity.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1], without any extensions, has difficulty in networks that contain Network Address Translators (NAT). SIP, out of necessity, breaks many of the guidelines described in RFC 3235 [2]. NAT traversal for SIP is especially problematic for the media streams, which generally flow from user agent to user agent.

To remedy this, RFC 3581 [3] defines a SIP extension for symmetric response routing, which allows SIP itself to traverse NAT. In order for the media streams to traverse NAT, Interactive Connectivity Establishment (ICE) [4] is used. ICE describes a methodology for NAT traversal for multimedia signaling protocols, such as SIP. It also defines some extensions to the Session Description Protocol (SDP) [5] for conveying additional data. ICE makes use of several protocols, namely the Simple Traversal of UDP Through NAT (STUN) [6] and Traversal Using Relay NAT [7], in order to operate.

This document contains a number of example deployment topologies and network configurations. For each, it shows how clients compliant to the above specifications will properly establish communications, and indeed, will do so using the optimal media path for that scenario. This document focuses on media streams that are carried over the Real Time Transport Protocol (RTP) [8]. In all cases, only RTP is shown and discussed, to simplify the discussion. RTCP related operations (generally STUN queries parallel to the RTP ones) are omitted.

2. Residential Users

In this scenario, a user has a broadband connection to the Internet, using a cable modem or DSL, for example. In order to provide security, or to run multiple machines, the user has purchased an off-the-shelf "DSL Router" as they are called. These devices, manufactured by companies such as Linksys, Netgear, 2wire, and Netopia, generally include a NAT, simple firewall, DHCP server and client, and a built in ethernet switch of some sort. The firewall generally allows all outgoing traffic, but disallows incoming traffic unless specific port forwarding or a DMZ host has been configured. The NAT treatment of UDP in these boxes varies. The most common types appear to be full-cone and restricted cone.

The user in this scenario wishes to use a communications service from a retail provider, such as net2phone or deltathree, for example. The connection between the user and the provider is through the cable modem or DSL, through the public Internet. The user may have multiple PCs in their home accessing this service, but they are not related in any way. This scenario also includes the case where its not a PC, but a standalone SIP phone. In this case, the provider might be providing some kind of second line VoIP service. This scenario is depicted in Figure 1.

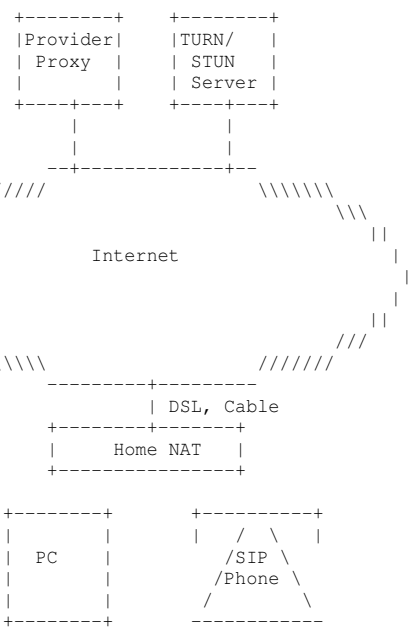
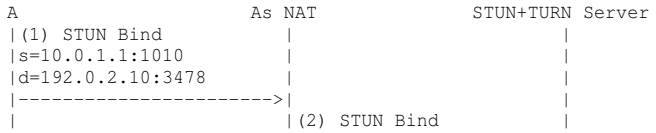


Figure 1: Residence with Single NAT

In this case, the provider administers a SIP proxy and a TURN/STUN server. This server is running STUN on the default port (3478) and TURN on port 5556.

2.1 Full Cone NAT



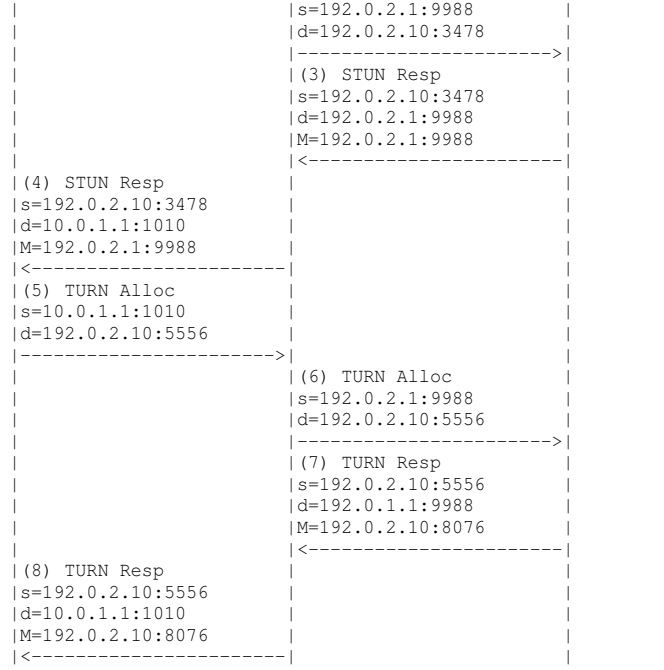


Figure 2: Message sequence for A's Unilateral Allocations

We first consider the case where two such residential users call each other, and both are using NATs of the full-cone variety. The caller follows the ICE algorithm. As such, it firsts allocates a pair of ports on its local interface for RTP and RTCP traffic (10.0.1.1:1010 and 10.0.1.1:1011). As shown in Figure 2, the client issues a STUN request from the RTP port (message 1), which passes through the NAT on its way to the STUN server. In the figure, the "s=" indicates the source transport address of the message, and "d=" indicates the destination transport address. The NAT translates the 10.0.1.1:1010 to 192.0.2.1:9988, and this request arrives at the STUN server (message 2). The STUN server copies the source address into the MAPPED-ADDRESS field in the STUN response (the M= line in message 3), and this passes through the NAT, back to the client. The client now

has a STUN derived transport address of 192.2.0.1:9988. Thought not show, the client will follow a similar process to obtain a STUN derived transport address for RTCP. However, this address will frequently not occupy an adjacent port to the RTP.

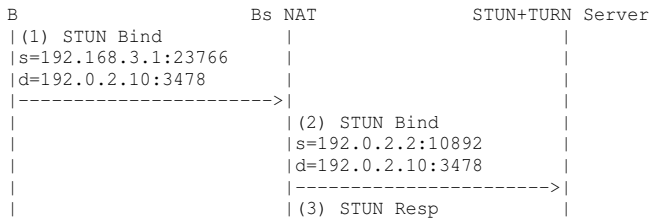
Next, the client follows a similar process to obtain a TURN port for RTP (messages 5-8). The TURN requests are also sent from the same local transport address. Note, however, that the TURN derived transport addresses for RTP (192.0.2.10:8076) and RTCP will be on adjacent ports. This is because the TURN pre-allocation procedure was used in the TURN request for the RTP port (message 5).

The client prioritizes these addresses, choosing the local interface address with priority 1.0, the STUN address with priority 0.8, and the TURN address with priority 0.4. From this, it generates an offer that looks like this:

```
v=0
o=alice 2890844730 2890844731 IN IP4 host.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8076 RTP/AVP 0
a=alt:1 1.0 : user 9kksj== 10.0.1.1 1010
a=alt:2 0.8 : user1 9kksk== 192.0.2.1 9988 192.0.2.1 9990
a=alt:3 0.4 : user2 9kksl== 192.0.2.10 8076
```

Figure 3: A's Offer

Note how the TURN derived transport address is used in the m and c lines, since this is the address with the highest probability of working with a non-ICE peer. That address is also included in the list of alteratives (with ID 3). Also note that because the STUN derived transport address for RTP and RTCP were not adjacent, two transport addresses are provided for alternate 2.



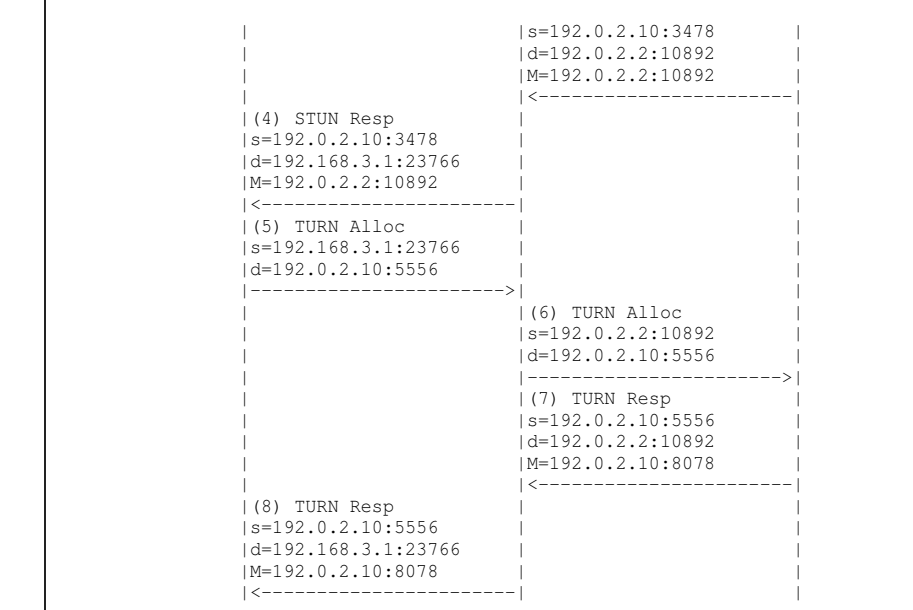
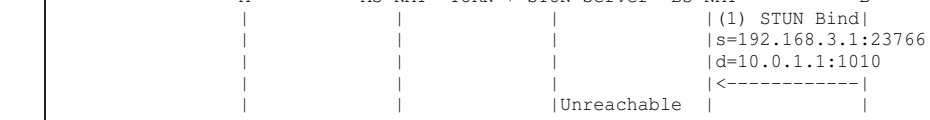


Figure 4: Message sequence for B's Unilateral Allocations

This offer arrives at the called party, user B. User B is also behind a full-cone NAT, and is using the 192.168/16 private address space internally. It happens to be using the same service provider as A, and is therefore using the same TURN server, at 192.0.2.10:5556. User B follows the same set of procedures followed by user A. It uses local interfaces, STUN, and TURN, and obtains a set of transport addresses that it can use. This process is shown in Figure 4. This process differs from that of Figure 2 only in the actual addresses and ports used and obtained.



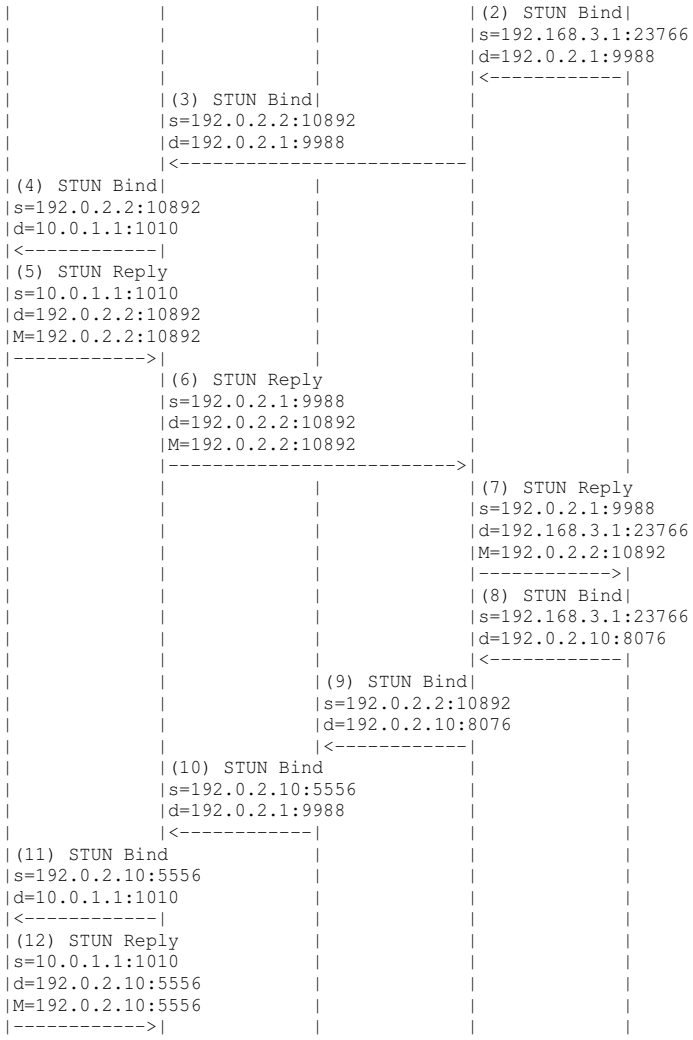


Figure 5: B's Connectivity Checks

While B's phone is ringing, B's user agent uses STUN to test connectivity from its local transport address pair (192.168.3.1:23766 and 192.168.3.1:23767) to the three alternates listed in the offer. The flow for that is shown in Figure 5. This flow, and the discussions, only consider the RTP transport addresses. The procedures would all be identical for RTCP. First, B tests connectivity to the alternate with ID 1, which is 10.0.1.1:1010. It does so by attempting to send a STUN request to this address (message 1). Of course, this is a private address, and not in the same network as B. Therefore, it is unreachable, and no STUN response is received.

In parallel, B tests connectivity to the alternate with ID 2, which is 192.0.2.1:9988. To do this, it sends a STUN request to that address. It sends it with a source address equal to its local transport address; the same one that it used to send the previous TURN and STUN packets (192.168.3.1:23766). This request (message 2) arrives at the NAT. Since the NAT is full cone, and since this address has an existing binding, the NAT translates the source address to that existing binding, 192.0.2.2:10892. This request (message 3) continues onwards to A's NAT. Since A's NAT is also full cone, the existing binding for 192.0.2.1:9988 is used, and the destination address is translated to 10.0.1.1:1010 and then forwarded towards A (message 4). A receives this. It verifies the username and password, and then generates a response. The response contains a MAPPED-ADDRESS equal to the source address seen in the STUN request (192.0.2.2:10892). It passes back through A's NAT (message 5), through B's NAT (message 6), and back to B (message 7).

B examines the MAPPED-ADDRESS in the STUN response. Its

192.0.2.2:10892. However, this is not a new address. B is already aware of this address as a result of its initial STUN Binding requests to the TURN/STUN server (Figure 4). As such, no additional addresses were learned.

In parallel with the tests against ID 2, B tests connectivity to the alternate with ID 3. This is the address A allocated through TURN. Of course, B does not know this. B sends a STUN request to this address (192.0.2.10:8076), and sends it from the same local transport address (192.168.3.1:23766) (message 8). The NAT, once again, translates the source address to 192.0.2.2:10892 (message 9). This is routed to the TURN server. The TURN server locks down the binding allocated to A, such that it will now begin relaying packets sent from A to 192.0.2.2:10892. The TURN server forwards the packet towards A (message 10). This reaches A's NAT, which translates the destination address based on the existing binding. The STUN request is then delivered to A (message 11). A verifies the username and password, and then generates a STUN response. This response contains the source address that the request came from. In this case, that source address is the public transport address of the TURN server (192.0.2.10:5556). This STUN response is relayed all the way back to B (messages 12-15).

B examines the MAPPED-ADDRESS in this STUN response. It's 192.0.2.10:5556, which is a new address. As a result, B has now obtained a peer derived STUN address. It adds this to its list of transport addresses. Its priority equals that of the address it was derived from - ID 3 - which has a qvalue of 0.4.

At some point, B picks up, and an answer is generated. The answer would look like this:

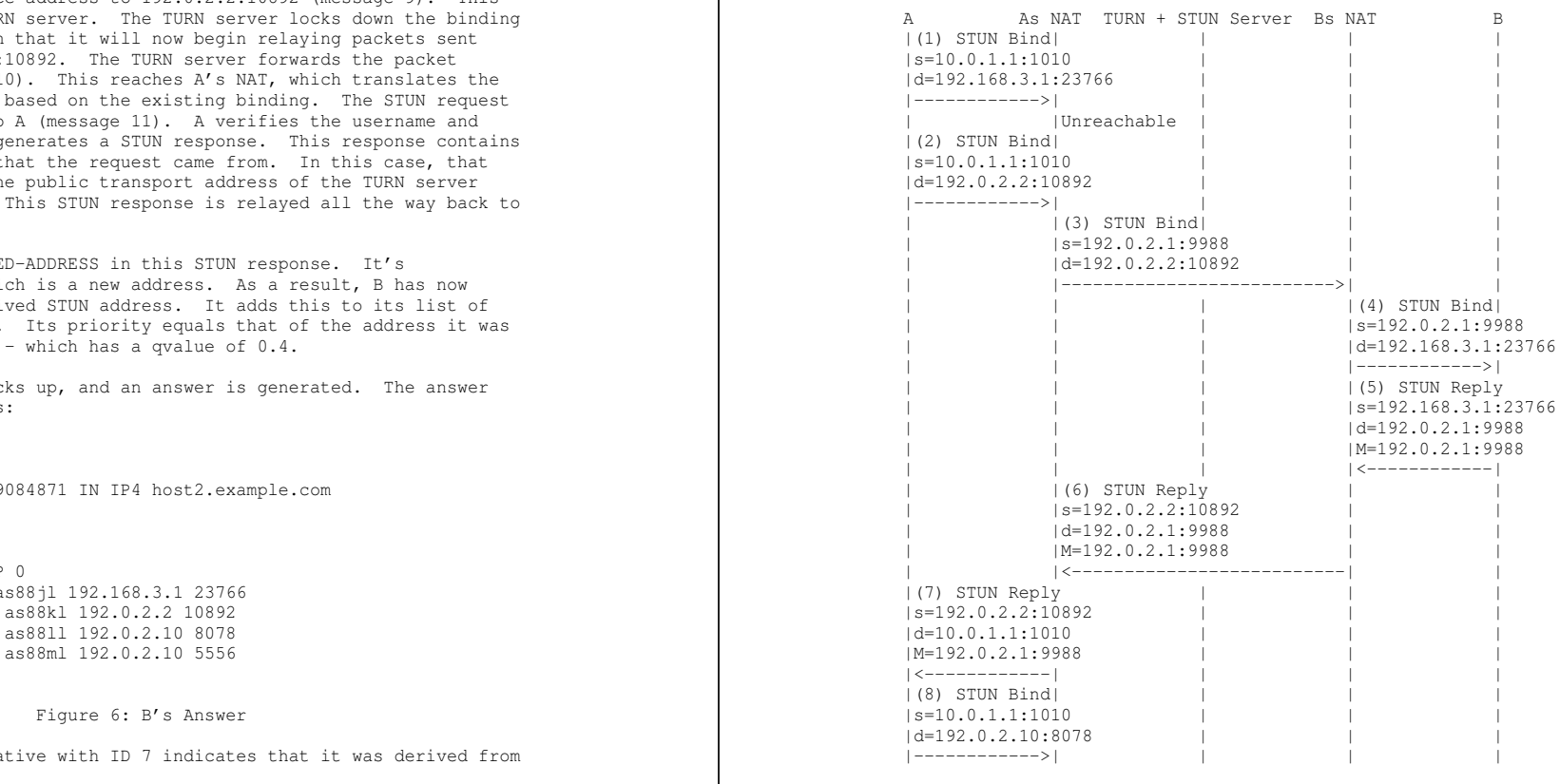
```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8078 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 192.168.3.1 23766
a=alt:5 0.8 : peer1 as88kl 192.0.2.2 10892
a=alt:6 0.4 : peer2 as88ll 192.0.2.10 8078
a=alt:7 0.4 3 peer3 as88ml 192.0.2.10 5556
```

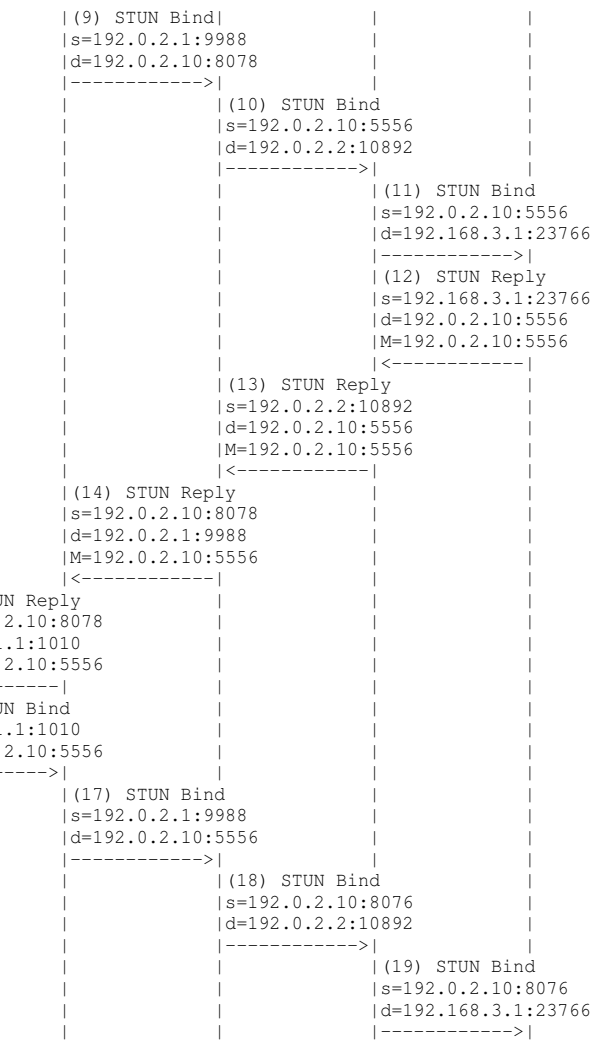
Figure 6: B's Answer

Note how the alternative with ID 7 indicates that it was derived from

the alternate with ID 3. Also, note that the four alternates use different IDs than the ones from the offer. This is for readability purposes only. The IDs are scoped to that specific agent, and there is no requirement that they do not use the same values.

This answer is sent to A. At the same time, B can send audio to A using the highest priority alternate that connectivity was established to. That is the alternate with ID 2, A's STUN derived transport address.





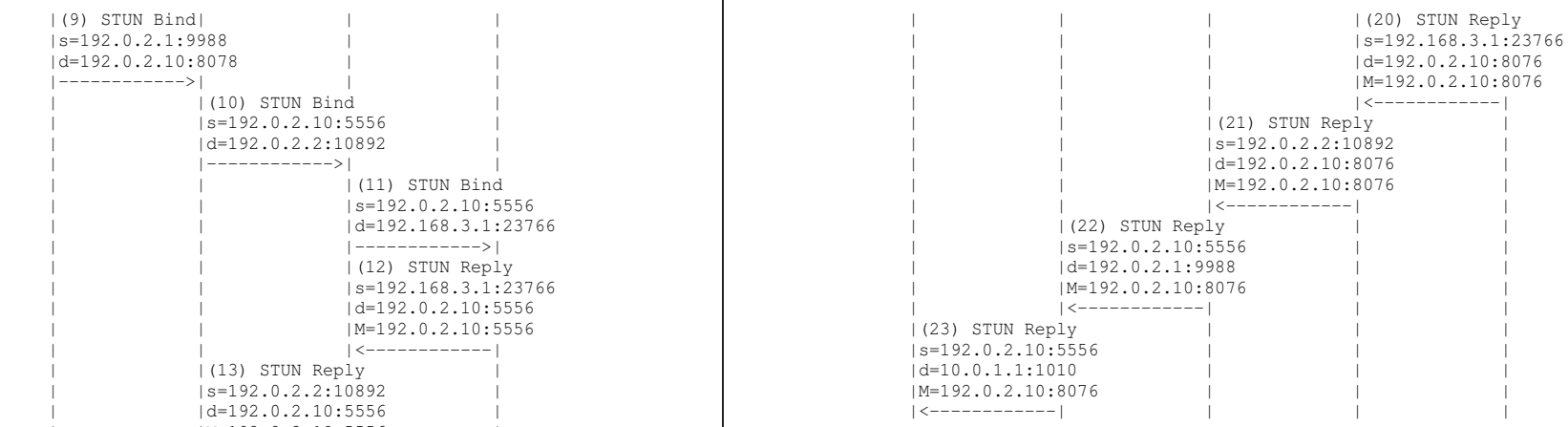


Figure 7: A's Connectivity Checks

When the answer arrives at A, A performs similar connectivity checks, shown in Figure 7. Each connectivity check is a STUN request sent from its local transport address (10.0.1.1:1010). The first is to the alternate with ID 4, which is 192.168.3.1:23766. The STUN request to this address (message 1) fails, since this is an unreachable private address. The second check is to the alternate with ID 5 (192.0.2.2:10892), which is the public address for B obtained as a result of STUN requests to the network server. Messages 2-7 represent the flow for this case. It is similar to the sequence in Figure 5 messages 2-7, differing only in the IP addresses. The result of this check provides a peer derived transport address of 192.0.2.1:9988. A already knows this address. The third connectivity check is to the alternate with ID 6 (192.0.2.10:8078). This represents A's TURN derived transport address. Messages 8-15 represent the check for this address, and they are also similar to messages 8-15 of Figure 5. This check provides A with a peer derived transport address of 192.0.2.10:5556. This represents a new address for A. It has a priority equal to the address it was derived from, which is 0.4.

The final connectivity check is to the alternate with ID 7 (192.0.2.10 5556). The SDP indicates that this address itself is a peer derived transport address. It was derived from A's transport address with ID 3, which is 192.0.2.10:8076, its TURN derived transport address. Because of that, the STUN request is sent from

the local transport address that 192.0.2.10:8076 was derived from. This local address is 10.0.1.1:1010. The message sequence for this check is represented by messages 16-23 of Figure 7. The STUN request is sent with a source address of 10.0.1.1:1010, to 192.0.2.10:5556. This is the well-known address of the TURN relay. This message passes through the NAT, and the source address is translated to A's public address, 192.0.2.1:9988 (message 17). Note that this same public address is used for all requests sent from 10.0.1.1:1010 because the NAT is full-cone. This arrives at the TURN server. The TURN server associates this message (which is just an arbitrary UDP packet as far as the TURN server is concerned) with the binding created for A. The peer in this case has been locked down. So, the packet is forwarded with a source address equal to the binding allocated to A (192.0.2.10:8076) and a destination address equal to the locked-down address (192.0.2.2:10892) (message 18). This arrives at B's NAT, where the destination address is translated to B's private address, 192.168.3.1:23766 (message 19). This arrives at B, which notes the source address in the STUN reply (192.0.2.10:8076). This reply is forwarded back to A (messages 20-23). From this, A sees a peer derived transport address of 192.0.2.10:8076. However, it already knows this address.

The result of the connectivity checks is that A determines it has connectivity to the alternates with IDs 5, 6 and 7. Of these, the one with ID 5 has the highest priority, and so this one is used to send media. Of course, A could have been sending media to B during these tests using the address in the m and c lines, which represents B's TURN derived transport address. Once the connectivity checks complete, A can switch to the one with ID 5, which is B's STUN derived transport address.

The connectivity checks also provided A with a new peer derived transport address - 192.0.2.10:5556 - with a priority of 0.4. However, A had received STUN requests on its alternates with IDs 2 and 3. The one with ID 2 (its STUN derived transport address) has higher priority than 0.4. So, A knows that generating a new ICE cycle to convey this address would not be useful. Thus, no new offer is sent. Indeed, since A had received a STUN request from B on its STUN derived transport address, A knows that its lower priority derived transport address is no longer needed. So, it is able to free up the TURN derived transport address a few seconds later. The same goes for B. Once it receives the STUN request to its TURN derived transport address (message 11 of Figure 7, it can free its TURN derived transport address.

In conclusion, the result in this case is that A and B will communicate with each other using their STUN derived transport addresses.

2.2 Symmetric NAT

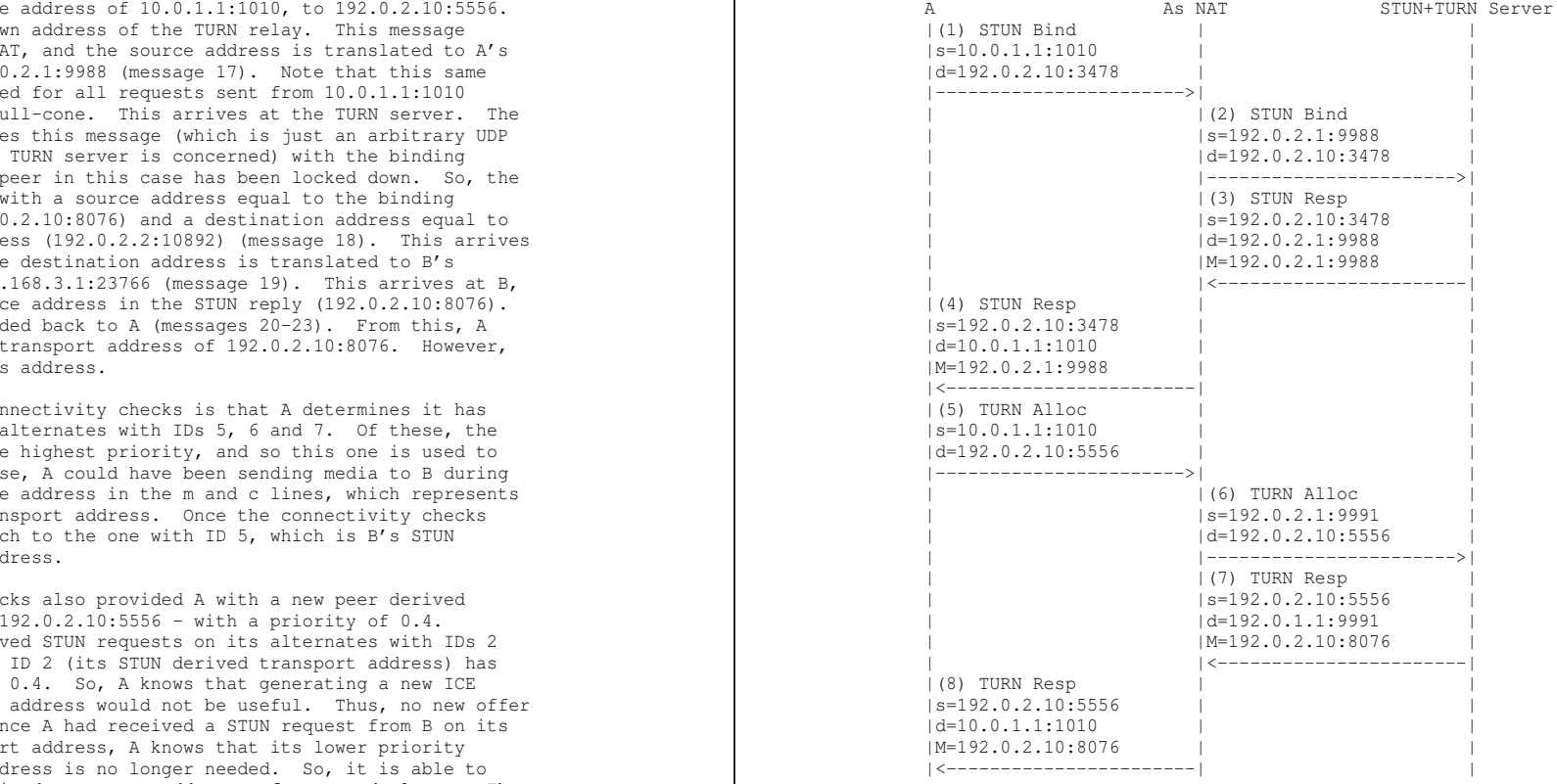


Figure 8: A's Unilateral Allocations

In this case, both residential users have symmetric NATs. The call starts again with A performing its unilateral allocations, as is shown in Figure 8. This message sequence is nearly identical to that of Figure 2. The only difference is that, because the NAT is

symmetric, different bindings are allocated for the two STUN and two TURN queries. A's discovers an identical set of addresses, however, and so generates the same offer as in Figure 3.

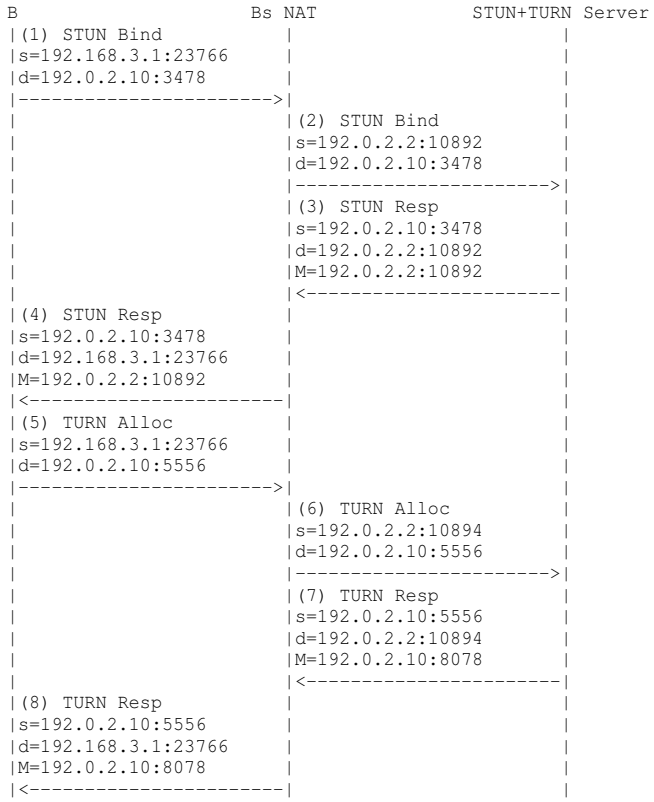
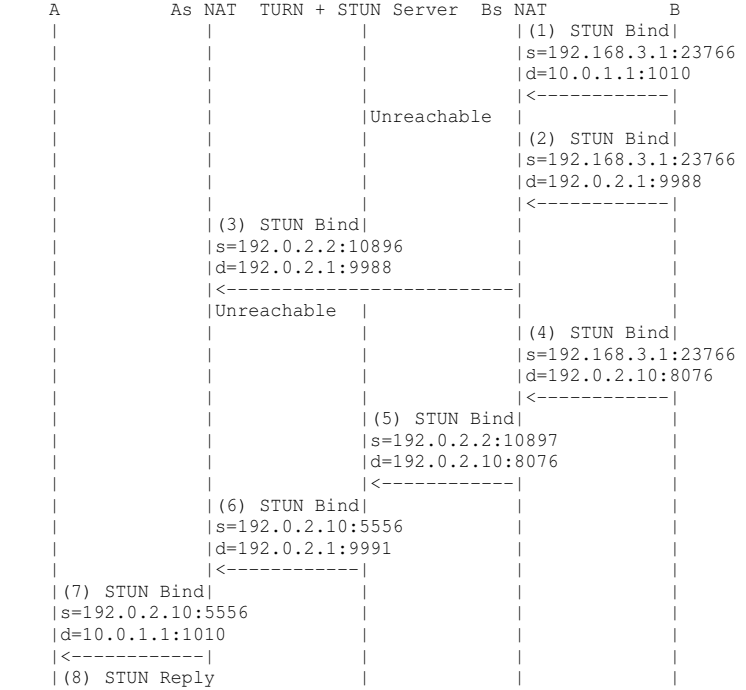


Figure 9: B's Unilateral Allocations

When B receives this offer, it performs its unilateral allocations. Like A's, these allocations (shown in Figure 9) are almost identical

to those in Figure 4. They differ in the same way - the NAT will allocate a different binding for each of the two STUN and two TURN queries. However, the set of derived transport address is the same. B now begins performing connectivity checks. These are shown in Figure 10. As in the previous case (Figure 5), the STUN request to 10.0.1.1:1010 fails. However, here, the STUN request to 192.0.2.1:9988 also fails. Thats because this packet arrives at A's NAT, and the NAT finds that the public transport address 192.0.2.1:9988 has been allocated, however, it was allocated when the client sent to 192.0.2.10:3478. Here, the source address is not 192.0.2.10:3478, and so the packet is discarded. The STUN request to 192.0.2.10:8076 does work, however. Thats because the TURN server sends the request from the same IP address and port that it received the original TURN allocation request on.



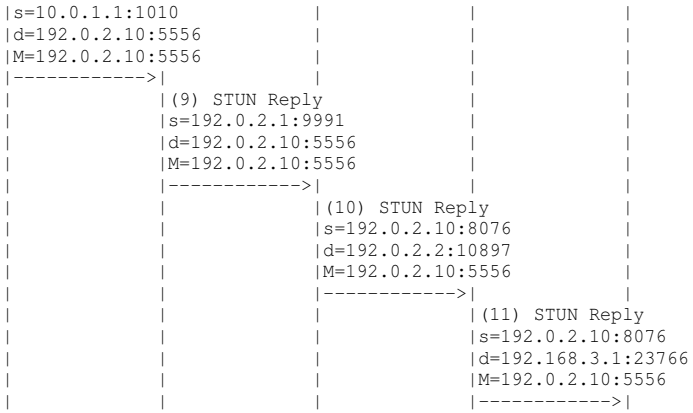
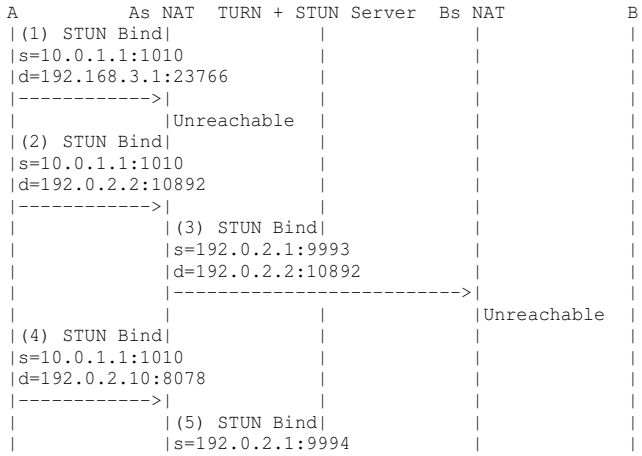


Figure 10: B's Connectivity Checks

B's answer to A is the same as in Figure 6. However, B has only established connectivity to A's TURN derived transport address, and so it sends media there.



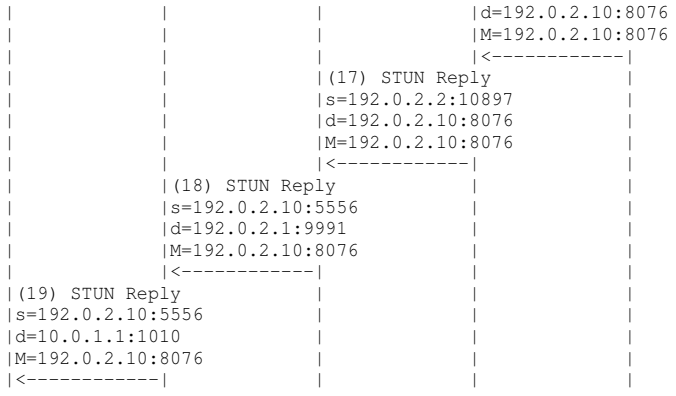


Figure 11: A's Connectivity Checks

When A gets the answer, it too performs its connectivity checks, as shown in Figure 11. As expected, the connectivity checks to B's private address and its STUN derived transport addresses fail. The checks to B's TURN derived transport address succeeds, as does the check to B's peer derived transport address. Both have a qvalue of 0.4. However, a peer-derived address is always preferred. So, A will send media to B using 192.0.2.10:5556, which will reach B as a result of the lock-down on its own TURN binding. As in the full-cone case, A won't bother to perform another offer with the new peer derived transport address it learned from message 19 (192.0.2.10:5556), since it knows that this is not of higher priority than ones that B has already connected to.

Once A connects to B's peer derived address (messages 12 to 19 in Figure 11), B knows that its equal priority TURN derived transport address won't be used, so it can free it.

OPEN ISSUE: The same argument can be made about A, in which case both sides would free their TURN addresses, and nothing works. Need to come up with a sane prioritization so it doesnt happen.

3. Basic Enterprise

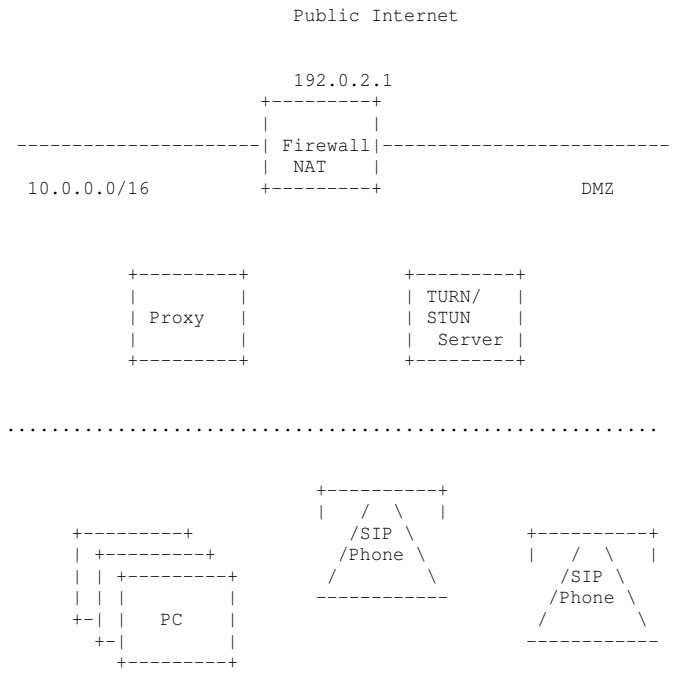


Figure 12: Enterprise Configuration

In this scenario, shown in Figure 12 there is an enterprise that wishes to deploy VoIP. The enterprise has a single site, and there is a firewall/NAT at the border to the public Internet. This NAT is symmetric. Internally, the enterprise is using 10.0.0.0/16. Behind the firewall, within the DMZ, is a TURN/STUN server and a SIP proxy. The firewall has been configured to allow incoming traffic to port 5060 to go to the SIP proxy. It has also allowed incoming UDP traffic on a specific port range to go to the TURN/STUN server. The

TURN server has an internal address of 10.0.1.10. This port range contains enough addresses to allow simultaneous conversations to cover the needs of the enterprise, but no more. External traffic sent to 192.0.2.1:8000 to 192.0.2.1:9000 is port forwarded to 10.0.1.10:8000 to 10.0.1.10:9000, respectively. That range is configured on the TURN/STUN server, so that the TURN server allocates addresses within this range.

Within the enterprise, PCs and hardphones are used for VoIP. All of them are configured to use the proxy and TURN/STUN server that is run by the enterprise. Furthermore, all of them are configured to use the TURN SEND mechanism for doing connectivity checks.

All call flows in this section only indicate RTP. The flows for RTCP are not shown.

3.1 Intra-Enterprise Call

In this section, we consider calls between two users in the same enterprise.

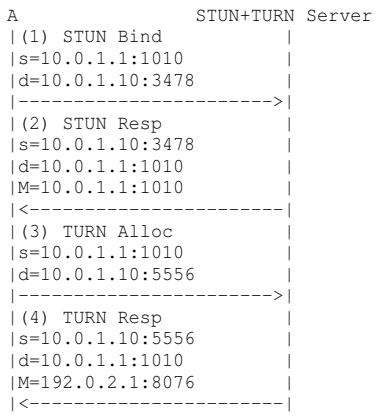


Figure 13: A's Unilateral Allocations

First, user A performs its unilateral allocations. This is shown in Figure 13. The STUN allocation does not yield a new address, but the TURN allocation, of course, does. The TURN address is publically routable. As a result, the offer from A to B has two addresses, as

shown in Figure 14.

```
v=0
o=alice 2890844730 2890844731 IN IP4 host.example.com
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 8076 RTP/AVP 0
a=alt:1 1.0 : user 9kksj== 10.0.1.1 1010
a=alt:2 0.5 : user1 9kksk== 192.0.2.1 8076
```

Figure 14: A's Offer

B receives this offer. It performs its own unilateral allocations, shown in Figure 15.

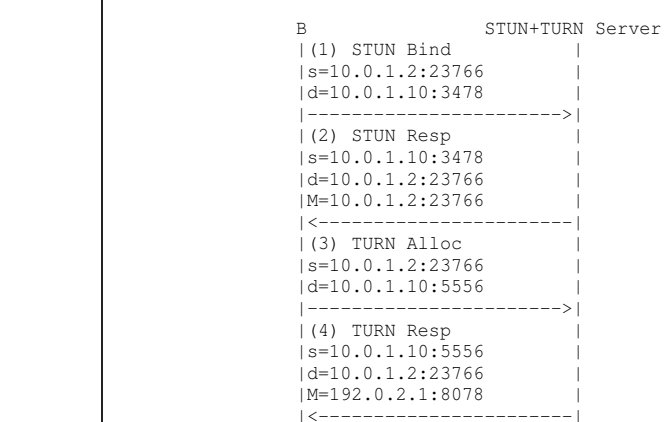
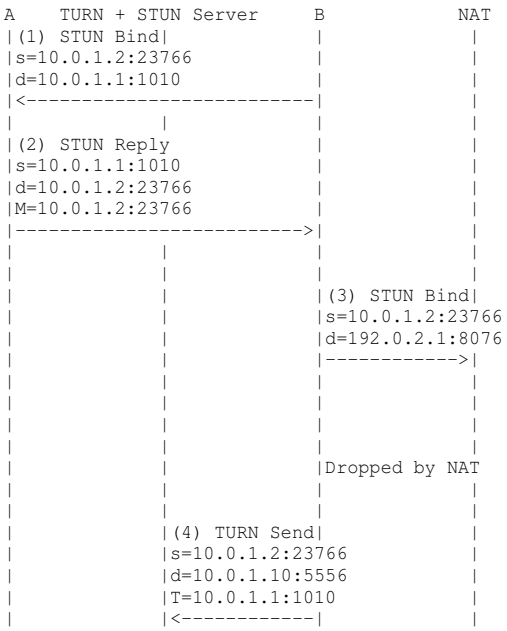


Figure 15: B's Unilateral Allocations

The STUN derived transport address equals its local transport address, so no additional addresses are obtained through that route. TURN provided B with a public address. Next, B performs connectivity checks against the two alternatives provided by A. These checks are shown in Figure 16. The connectivity check to the alternate with ID 1, A's local transport address, succeeds, since both users are within the same address realm. The connectivity to check to the alternate with ID 2, which is the TURN server address on the public Internet,

fails. This is because the NAT does not support receipt of requests from internal hosts that are targeted towards internal bindings. As a result, the STUN request is dropped by the NAT.

Because of its configuration, B also attempts to perform connectivity checks by sending STUN Bind requests though its TURN relay, using the TURN SEND command. As described in ICE, these connectivity checks need to be performed sequentially, not in parallel. B first attempts a send to deliver a STUN Bind request to A's local transport address (message 4). This is relayed by the TURN server to A, using the internal version of B's TURN derived transport address (10.0.1.10:8078) as the source address (message 5). This is the address that the NAT will translate 192.0.2.2:8078 into when it receives packets externally. A replies to this (message 6), reporting to B a new address, 10.0.1.10:8078. This is received by the TURN server, causing lock down to occur. The TURN server forwards this response back to B.



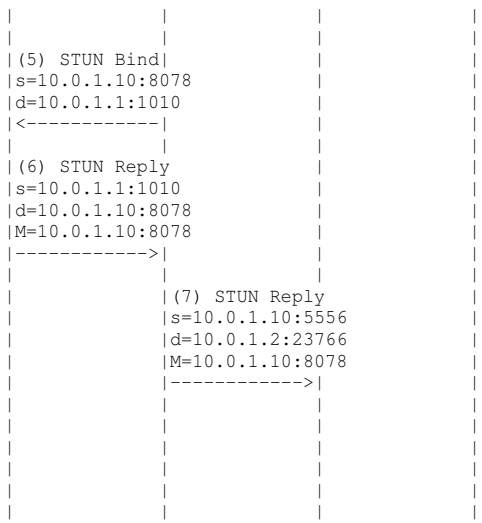


Figure 16: B's Connectivity Test

Based on this, B generates the answer shown in Figure 17. Since B has established connectivity to A's local transport address, it begins sending media there.

```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 8078 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 10.0.1.2 23766
a=alt:6 0.5 1 peer2 asjj8n 10.0.1.10 8078
a=alt:5 0.5 : peer1 as88kl 192.0.2.1 8078
```

Figure 17: B's Answer

Now, A performs its connectivity checks, shown in Figure 18. First, it checks for connectivity to B's local transport address (message 1). This connectivity check passes, and does not provide A with a

new address (message 2). Next, A checks for connectivity to 10.0.1.10:8078, the internal version of B's TURN derived transport address. This connectivity check (messages 3-6) also succeed, and provide A with a new peer derived transport address (10.0.1.10:5556). However, this address would have a lower priority (0.5) than that of one that B has already connected to (A's local transport address), and so A does not bother with another ICE cycle. The check to B's public TURN derived transport address fails (message 7). Since A discovers connectivity to a high priority transport address, it does not bother to perform its connectivity checks by relaying STUN messages through its TURN server. Both A and B can now free their TURN derived addresses, since both have established connectivity to higher priority addresses. The call proceeds with media flowing directly between A and B, as desired.

Note, however, that this call flow would not have worked if A supported ICE, but B didn't. Thats because the default TURN address will not work for internal clients. In enterprises where this is a concern, an alternate deployment, described in Section 4, works properly.

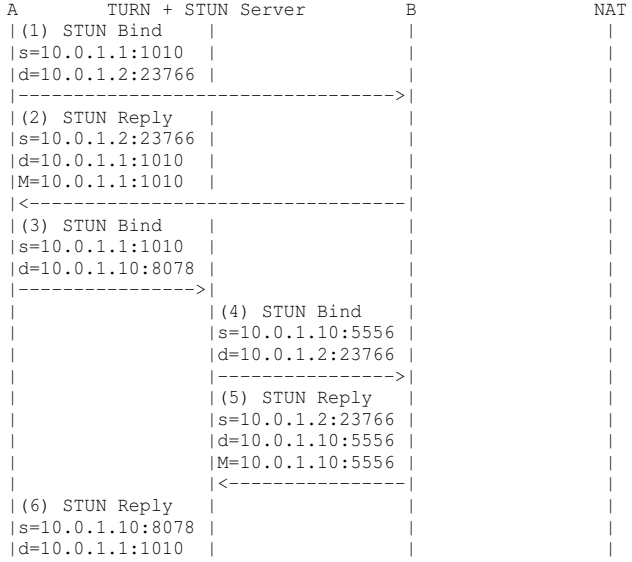


Figure 18: A's Connectivity Checks

3.2 Extra-Enterprise Call

In this case, user A within the enterprise calls some user B, not within the enterprise. B is connected to the Internet through a PSTN gateway, and as a result, appears as a UA on the public Internet. Presumably this is some gateway run by a third party termination provider that is being used by the enterprise. Furthermore, this gateway does not support ICE at all, and so will ignore the alt parameters in the SDP.

First, A performs its unilateral allocations. This proceeds identically as shown in Figure 13. It generates the same offer as shown in Figure 14. This gets routed to the called party on the public Internet. This party generates an answer. However, since the called party does not support ICE, the answer has no alt attributes. It has a single IP address and port listed in the c and m lines. As a result, the caller, A, needs to send media there. However, the enterprise policy prohibits outbound UDP traffic from end user devices. Thus, A has been configured to ensure outbound media flows through the TURN server. ICE would normally discover this, and media would flow that way. However, since ICE is not supported, it needs to be done explicitly by the client.

To accomplish this, A performs another, separate unilateral allocation to obtain another TURN address. It does not advertise this address to the called party. Instead, it issues a TURN SEND command to the IP address and port in the SDP answer. This send command contains the first RTP packet to send. From that point forward, A sends its media packets to the TURN server. The TURN server will forward those packets to the last address used in a SEND command, as long as lockdown has not occurred. Here, it will not, since the address learned from the TURN server is never advertised to any peers.

3.3 Inter-Enterprise

In this case, a user in one enterprise calls a user in another enterprise. In this configuration, media needs to flow through the TURN relays run by both enterprises, since the policies of both enterprises require this. We assume that B's enterprise is using 192.168/16 internally, and it has an external public IP address of 192.0.2.2. The TURN/STUN server is running on 192.168.1.10, using port 3478 for STUN and 5556 for TURN. Packets sent to 192.0.2.2:6500 to 192.0.2.2:6600 are forwarded to 192.168.1.10:6500 to 192.168.1.10:6600 respectively.

First, A performs its allocations. These are identical to the ones in Figure 13. The offer sent by A, as a result, is identical to Figure 14.

This call is received by B. B performs its allocations, shown in Figure 19. These are similar to those of Figure 15, differing only in the addresses used.

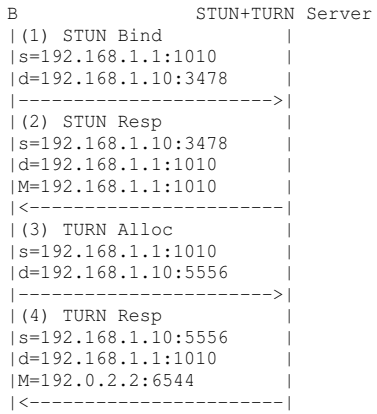
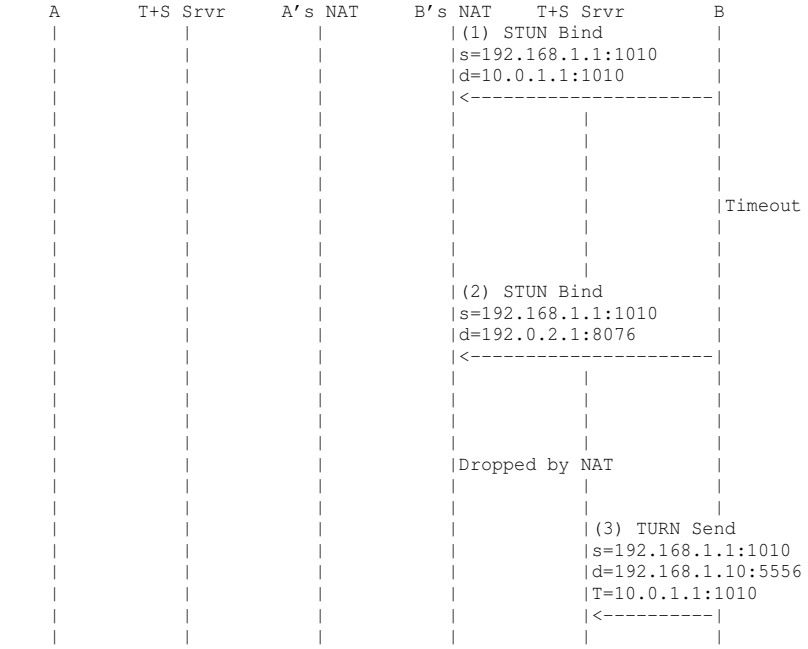


Figure 19: B's Unilateral Allocations

Next, B performs its connectivity checks, as shown Figure 20. First, B checks connectivity to A's local transport address (10.0.1.1:1010). This is unroutable within B's network, and so the STUN request is dropped by the routers in the network, and the check times out and fails. In parallel, B checks connectivity to A's TURN derived

transport address (192.0.2.1:8076). It sends a STUN Bind request to this address (message 2). This arrives at B's firewall/NAT. However, the firewall function does not allow outbound UDP packets from internal clients, and so the packet is dropped. This check also times out and fails. B also begins checking connectivity to A's two addresses by SENDing the STUN requests through its TURN server. First, B tries A's local transport address (message 3). This is relayed by the TURN server to 10.0.1.1:1010, which is dropped by the routers as well. Finally, B tries A's TURN derived transport address (message 4). This is successfully relayed all the way to A, as a result of the static bindings in place in A's and B's NATs. A sees a source address of 10.0.1.10:5556, which it reports back in the STUN reply. The STUN request (message 8) to A's TURN server locks down the binding, and the STUN reply (message 13) locks down the binding at B's TURN server. Based on the connectivity checks, B has learned a single new peer derived transport address, 10.0.1.10:5556.



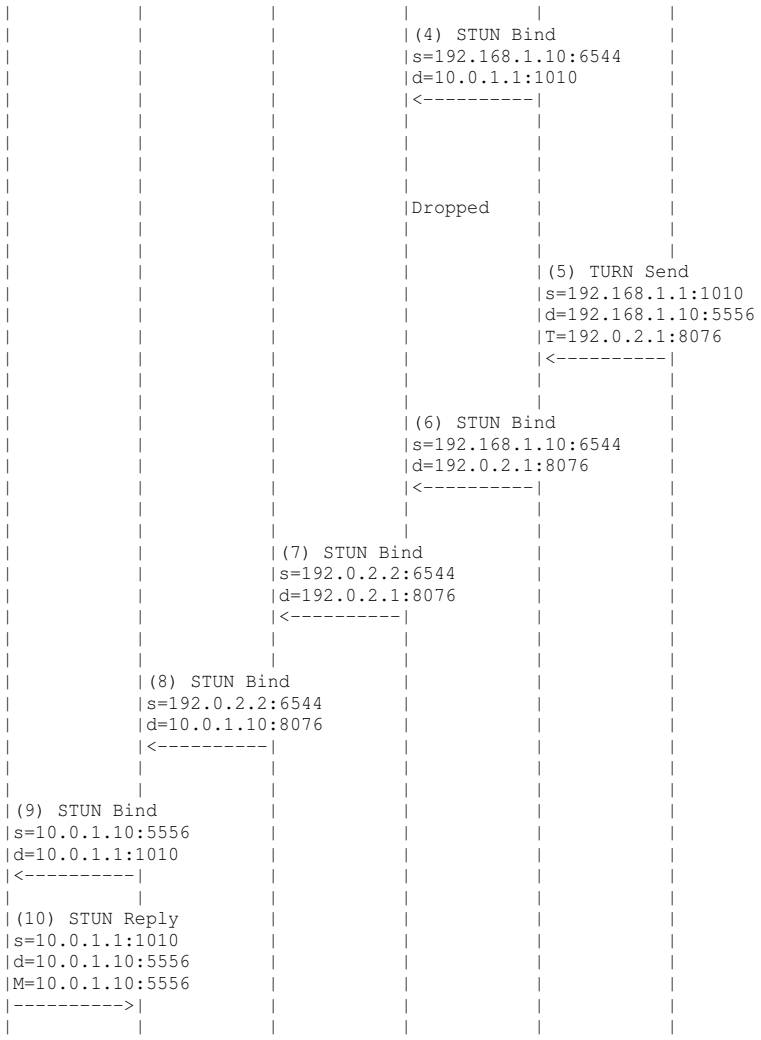
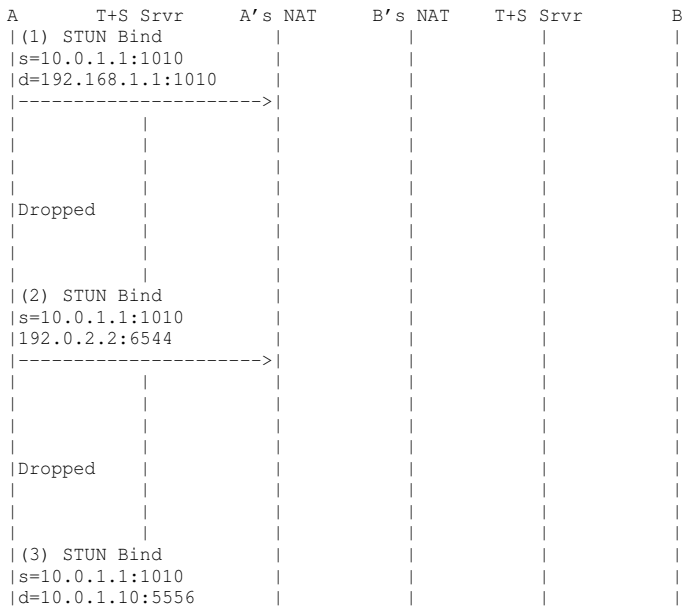


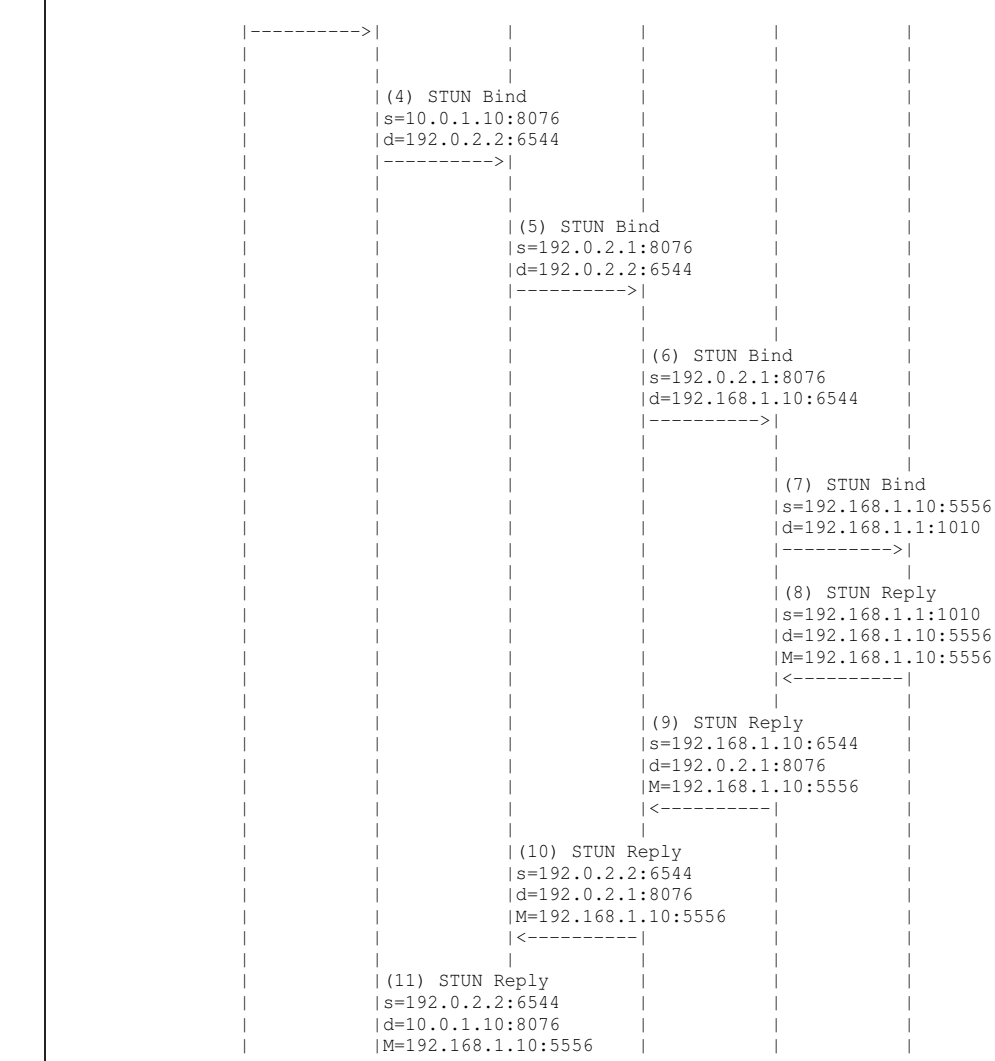
Figure 20: B's Connectivity Test

B's connectivity check showed that the only place where media can be sent is through its relay. Since the binding has been locked down, B knows it can just send raw media packets to the relay, which will be forwarded appropriately. As such, it begins sending media through the relay pairs. B also generates its answer:

```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 6544 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 192.168.1.1 1010
a=alt:5 0.5 : peer1 as88kl 192.0.2.2 6544
a=alt:6 0.5 2 peer3 hh8sdl 10.0.1.10 5556
```

Now, A performs its connectivity checks, which are shown in Figure 22. These checks are similar to those of Figure 20. A's TURN server relays the STUN request towards B's TURN server because of the lock-down from B;s connectivity test. A's test reveals connectivity to 10.0.1.10:5556, which is B's peer derived address. Since connectivity was established there, A does not bother doing connectivity checks by SENDing STUN requests through its TURN server. The media proceeds to flow through both relays.





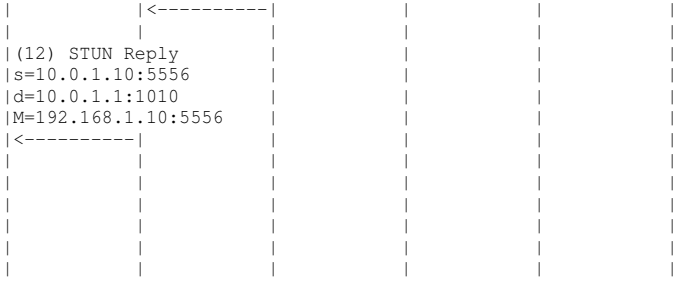
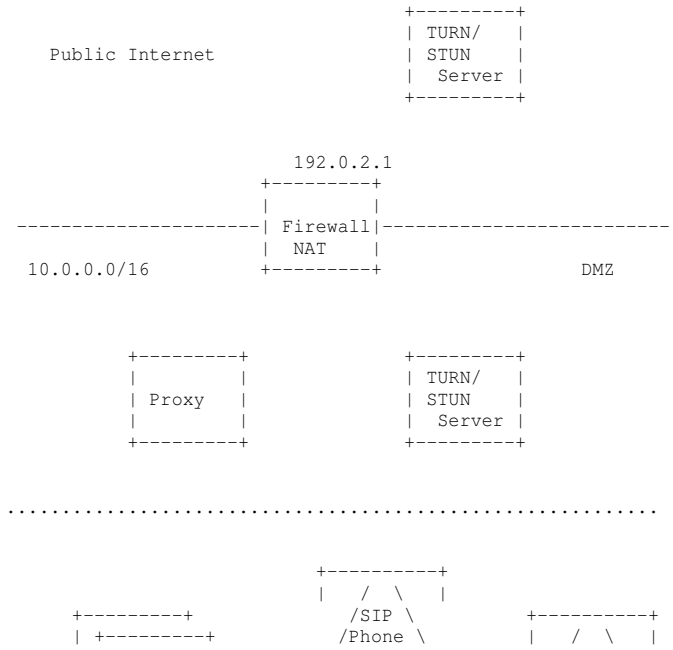


Figure 22: A's Connectivity Checks

4. Advanced Enterprise

The network of Section 3 describes a basic enterprise. It requires the enterprise to configure port forwarding on a range of external addresses, forwarding them to the internal TURN server. It also requires that ICE be deployed within the whole enterprise, since the default address won't work when talking to non-ICE clients within the enterprise.

A more complex network design can be used in enterprises that refuse to enable port forwarding/static bindings, and for which a heterogeneous internal network is expected. The design of this network is shown in Figure 23



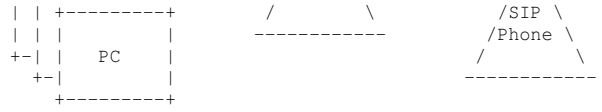


Figure 23: Enterprise Configuration

In this network, there are two TURN servers. There is one internal to the firewall, and one external. Clients only contact the internal one directly. This TURN server authenticates the client, and then obtains the public binding by sending a TURN request to the external TURN server. The external TURN server returns a public address, which is forwarded to the client by the internal TURN server. The TURN query from the internal to external server creates a NAT binding in the enterprise NAT, and therefore, static bindings are no longer required. Authentication is done by the internal TURN server so that the external server does not need to contact an internal database; all database access is kept internal. The external TURN server still authenticates the TURN query, but the authentication is done using a configured username and password, configured into both the external and internal servers. For security, that username and password can be highly randomized and altered periodically - it is not used by end users, but rather by network equipment.

TODO: Add call flows.

5. Centrex

In a centrex scenario, a third party provider owns and operates the SIP and TURN/STUN servers. The enterprise merely changes their firewall configuration to allow SIP traffic out to port 5060 to the provider's SIP proxy, and to allow TURN traffic out to port 5556 and 3478, on the provider's TURN/STUN server. The corporate NAT is symmetric. The TURN/STUN server runs on 192.0.2.10. This scenario is shown in Figure 24.

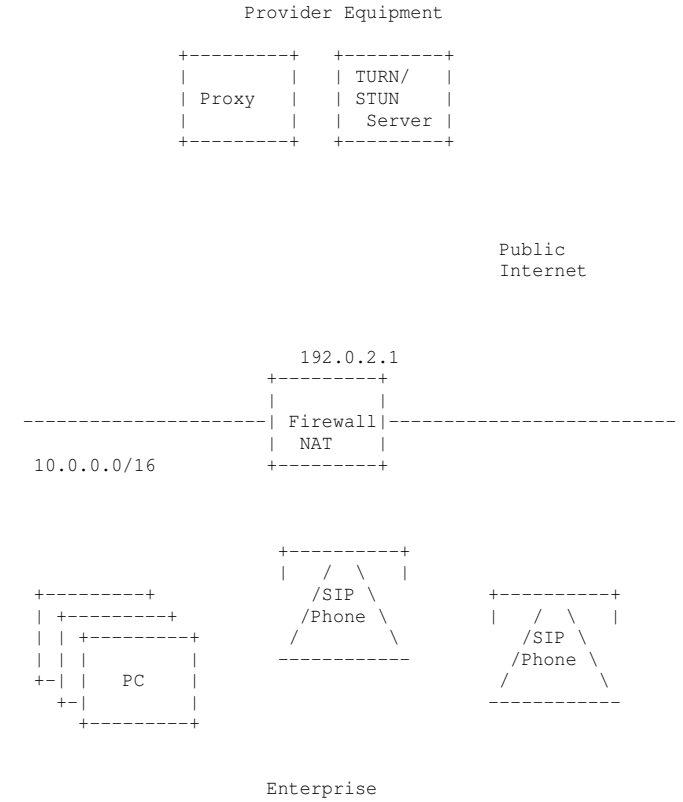
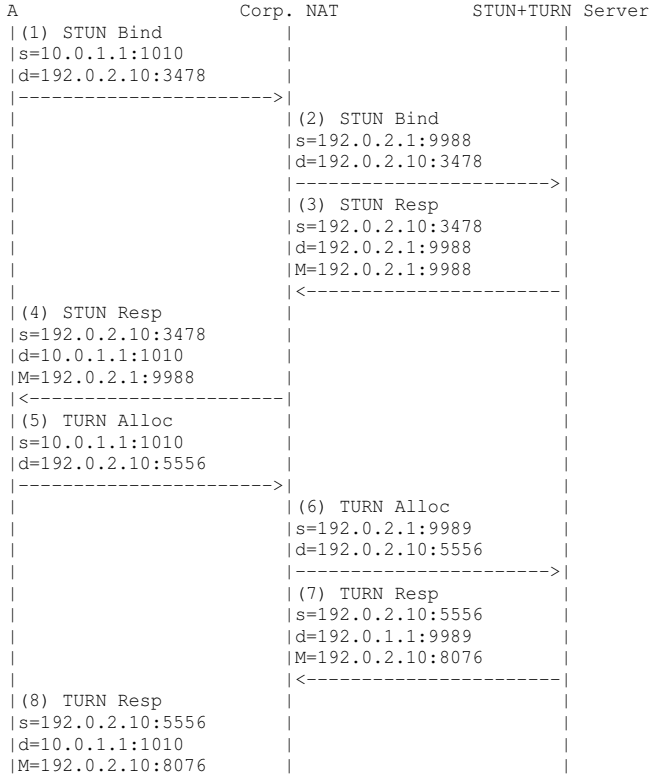


Figure 24: Centrex Configuration

5.1 Intra-Enterprise Call

In this scenario, user A calls user B. Both are within the enterprise. First, A performs its unilateral allocations. These are shown in Figure 25. These yield a STUN derived transport address and a TURN derived transport address. A sends these in the offer shown in Figure 26.



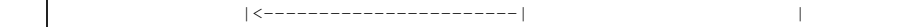
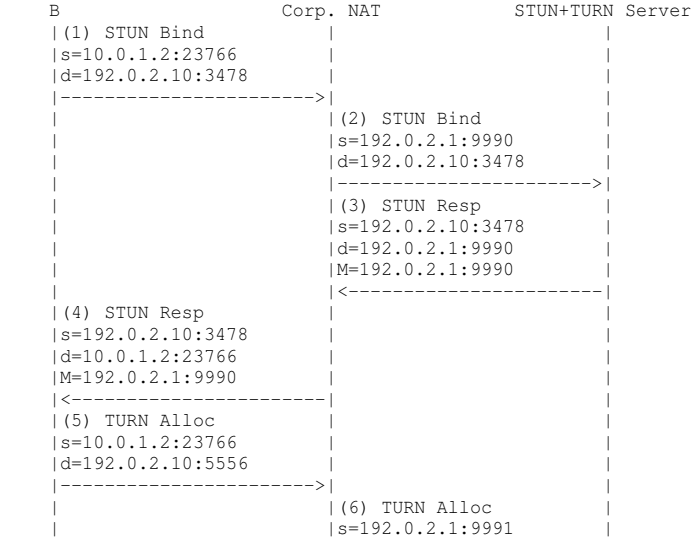


Figure 25: A's Unilateral Allocations

```
v=0
o=alice 2890844730 2890844731 IN IP4 host.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8076 RTP/AVP 0
a=alt:1 1.0 : user 9kksj== 10.0.1.1 1010
a=alt:2 0.5 : user1 9kksk== 192.0.2.1 9988
a=alt:3 0.4 : user2 9kksl== 192.0.2.10 8076
```

Figure 26: A's Offer

This offer is received by B. B performs its unilateral allocations, shown in Figure 27. These yield a STUN derived and TURN derived transport address.



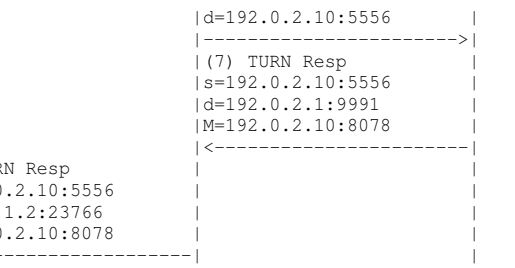
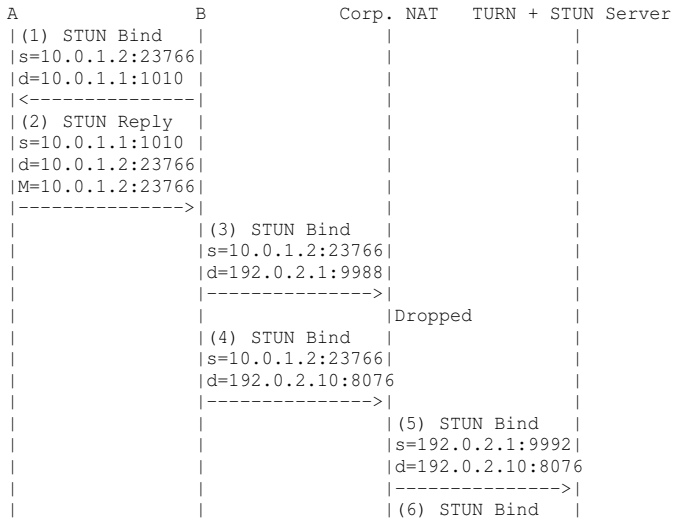


Figure 27: B's Unilateral Allocations

Now, B begins its connectivity checks, as shown in Figure 28. The first check (message 1), to A's local transport address, 10.0.1.1:1010, succeeds, since A and B are behind the same NAT. The second check, to A's STUN derived transport address, fails, since the enterprise NAT won't turn around packets. The third check, to A's TURN derived transport address, 192.0.2.10:8076, also succeeds, and yields B a new peer derived transport address, 192.0.2.10:5556.



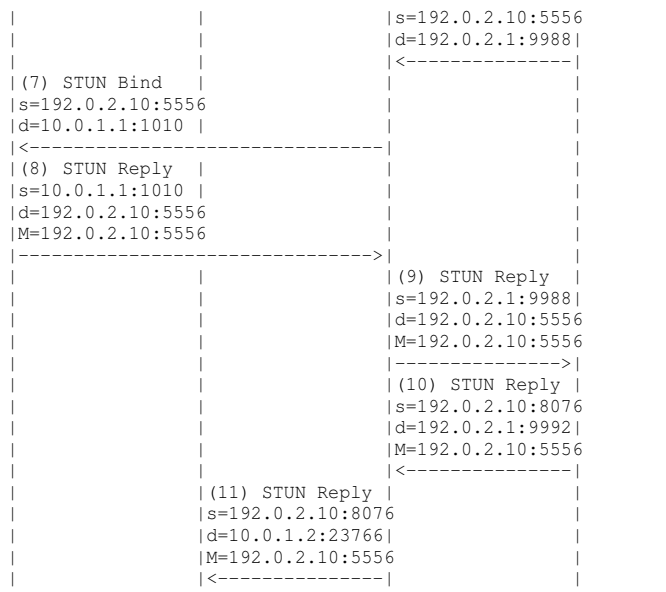


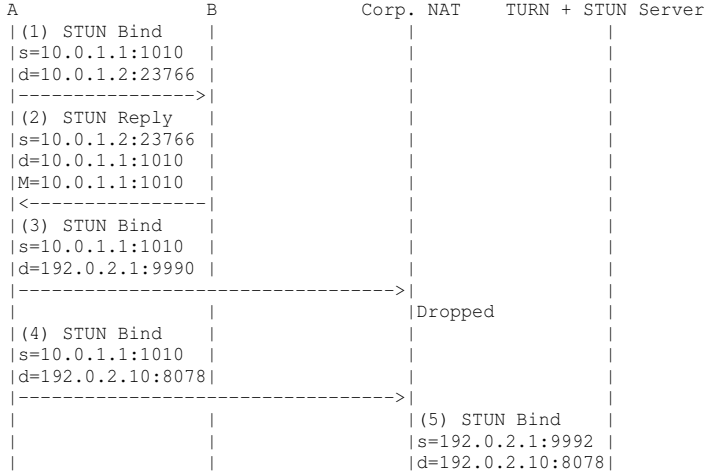
Figure 28: B's Connectivity Checks

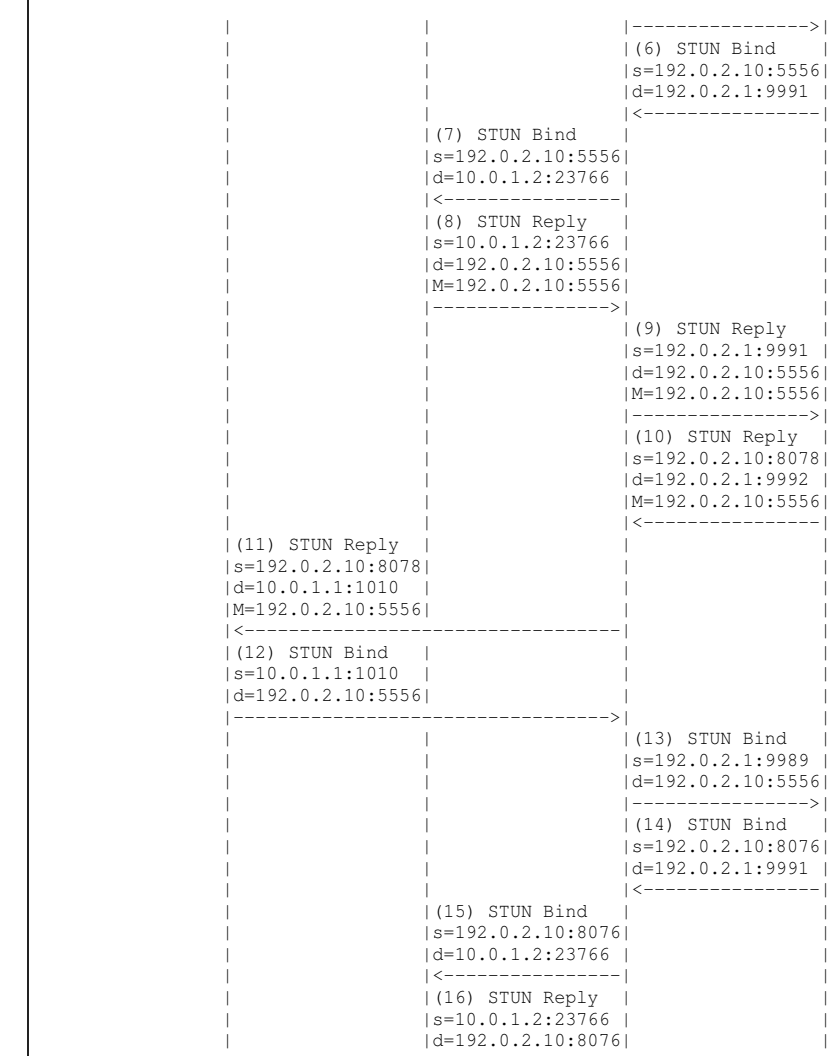
B can now send media to A directly. It also generates an answer, shown in Figure 29.

```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8078 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 10.0.1.2 23766
a=alt:5 0.8 : peer1 as88kl 192.0.2.1 9990
a=alt:6 0.4 : peer2 as88ll 192.0.2.10 8078
a=alt:7 0.4 : peer3 as88ml 192.0.2.10 5556
```

Figure 29: B's Answer

This arrives at A. A is able to send media immediately to B using the default, 192.0.2.10:8078. It also starts its connectivity checks to find a better choice. These checks are shown in Figure 30. As expected, the check for connectivity to 10.0.1.2:23766 succeeds, representing the highest priority address. The check to 192.0.2.1:9990 fails, because the NAT won't turn around internal packets. The checks to 192.0.2.10:8078 and 192.0.2.10:5556 succeed, and the former resuls in a peer-derived transport address of 192.0.2.10:5556. However, A knows that B has already connected to a higher priority address, so it doesn't bother with an additional offer/answer exchange.





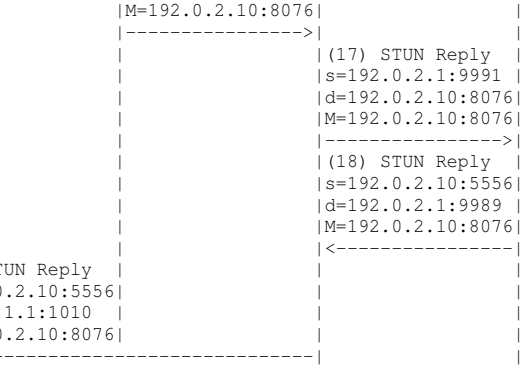


Figure 30: A's Connectivity Checks

The conclusion is that A and B communicate directly, without using the provider's relay. They can proceed to de-allocate the TURN addresses once the call is active.

6. An IPv6 Network with a pool of IPv4 addresses

Figure 31

This example deals with a network of IPv6 SIP user agents that has a NAT with a pool of public IPv4 addresses, as shown in Figure 31. The NAT advertises the prefix PREFIX::/96 in the IPv6 network, so all

packets addresses to that PREFIX are routed to the NAT, as described in RFC 2766 [9]. The IPv6 SIP user agents of this IPv6 network need to communicate with users on the IPv4 Internet and with residential users behind a NAT (i.e., with private IPv4 addresses), even if those residential users do not have access to any STUN or TURN servers. It is assumed, though, that the residential users can run STUN servers on their ports.

For a particular session, a given IPv6 SIP user agent can obtain the services from the NAT. The NAT receives IP packets from the IPv6 SIP terminal on an IPv6 address and forwards them to the peer's IPv4 address (as seen from the NAT). It also receives packets from the peer on an IPv4 address and forwards them to the IPv6 address of the IPv6 SIP user agent.

This scenario is different from the residential user scenario described in Section 2 because the IPv6 terminal needs to communicate with the NAT to obtain a public IPv4 address to place in its offer and answers. This is because residential users would not understand IPv6 addresses in the SDP. The way the IPv6 SIP user agent obtains this IPv4 address is outside the scope of this document.

The 3G IP Multimedia Subsystem (IMS) has the characteristics just described. A solution that allows IPv6 IMS terminals to communicate with Internet users where the terminals obtain the public IPv4 address from the NAT using session policies is described in [10].

6.1 Initial Offer Generated by the IPv6 SIP User Agent

In this example, an IPv6 SIP user agent sends an offer to a residential user that is located behind a NAT. Before generating an offer, the IPv6 SIP user agent obtains a public IPv4 address from the NAT. The IPv6 SIP user agent groups both addresses (its IPv6 address and the public IPv4 address that it just obtained) using the IPV semantics [11] and places them in its offer, which is shown in Figure 32.

```
v=0
o=bob 280744730 28977631 IN IP6 host.example.com
s=
t=0 0
a=group:IPV 1 2
m=audio 6886 RTP/AVP 0
c=IN IP6 2001:056D::112E:144A:1E24
a=mid:1
m=audio 22334 RTP/AVP 0
c=IN IP4 192.0.0.1
```



```
a=mid:2
a=alt:1 1.0 : user 9kksj== 192.0.0.1 22334
```

Figure 32

When the residential user receives the offer in Figure 32, it uses STUN to obtain new addresses to place in its answer, as shown in Figure 33. The IPv6 SIP user agent responds to the residential user's STUN Bind messages with a STUN reply. This STUN reply carries a new address (192.0.1.1:2000), which the residential user places in its answer, shown in Figure 34. The answer indicates that this address has been derived from the alternative number 1 in the offer. Since the residential user does not support IPv6, it sets the port number of the media stream with the IPv6 address to zero.

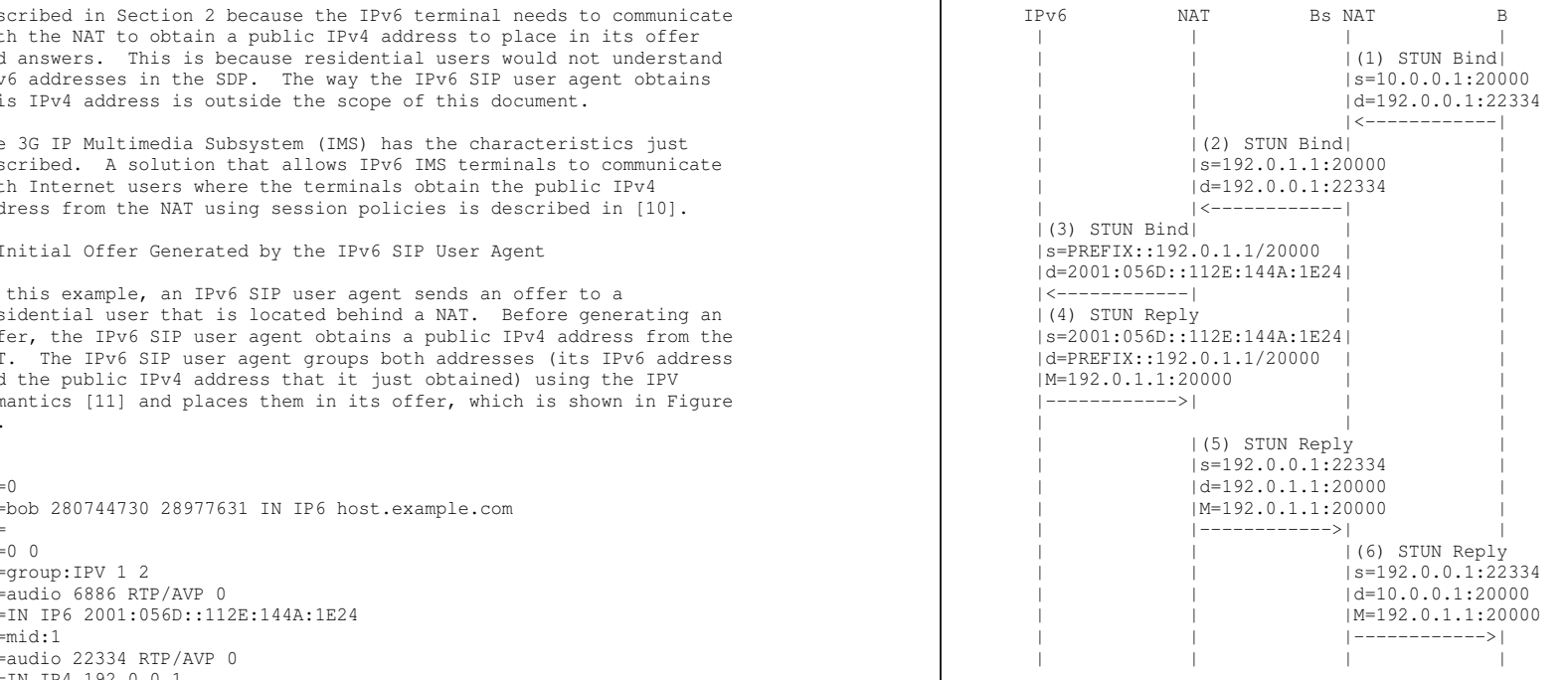


Figure 33

```
v=0
o=alice 280756730 28956631 IN IP4 host.example2.com
s=
t=0 0
m=audio 0 RTP/AVP 0
m=audio 20000 RTP/AVP 0
c=IN IP4 10.0.0.1
a=alt:2 1.0 : peer as88jl 10.0.0.1 20000
a=alt:3 0.8 1 peer as88kl 192.0.1.1 20000
```

Figure 34

When the IPv6 SIP user agent receives the answer, it uses STUN to check both addresses, 10.0.0.1:20000 and 192.0.1.1:20000. When it does so, it discovers that 10.0.0.1:20000 is unreachable and that 192.0.1.1:2000 can be used to send media to the peer.

Alternatively, the IPv6 SIP user agent could take advantage of knowing that its own IPv4 address is public and deduct which peer address to use without using STUN. If the answer contains an address which was derived from an alternative in the offer, that address will have best connectivity. If the answer does not contain any derived address, it means that the peer has a local public IPv4 address, which will be the alternative with highest priority in the answer.

6.2 Initial Offer Generated by the Residential User

In this example, a residential user that is located behind a NAT sends an offer to the IPv6 SIP user agent. The residential user places its local IPv4 address in the offer, as shown in Figure 35.

```
v=0
o=alice 280756730 28956631 IN IP4 host.example2.com
s=
t=0 0
m=audio 20000 RTP/AVP 0
c=IN IP4 10.0.0.1
a=alt:1 1.0 : peer as88jl 10.0.0.1 20000
```

Figure 35

The IPv6 SIP user agent uses STUN towards 10.0.0.1, which is unreachable. Consequently, no new addresses are discovered.

Alternatively, the IPv6 SIP user agent can skip using STUN at this point, since it knows that its NAT provides public IPv4 addresses. It does not really have any need to discover any new addresses.

The IPv6 SIP user agent places a public IPv4 address that it obtains from the NAT in its answer, as shown in Figure 36.

```
v=0
o=bob 280744730 28944631 IN IP6 host.example.com
s=
t=0 0
m=audio 22334 RTP/AVP 0
c=IN IP4 192.0.0.1
a=alt:2 1.0 : user 9kksj== 192.0.0.1 22334
```

Figure 36

When the residential user receives the answer from the IPv6 SIP user agent, it uses STUN to discover its IP address as seen by its peer (192.0.1.1:20000). The call flow is idential to the one shown in Figure 33. Then, it sends a new offer, which is shown in Figure 37.

```
v=0
o=alice 280756730 28956632 IN IP4 host.example2.com
s=
t=0 0
m=audio 20000 RTP/AVP 0
c=IN IP4 10.0.0.1
a=alt:1 1.0 : peer as88jl 10.0.0.1 20000
a=alt:3 0.8 2 peer as88kl 192.0.1.1 20000
```

Figure 37

When the IPv6 SIP user agent receives the offer in Figure 37, it uses STUN to check both addresses, 10.0.0.1:20000 and 192.0.1.1:20000. When it does so, it discovers that 10.0.0.1:20000 is unreachable and that 192.0.1.1:2000 can be used to send media to the peer.

Alternatively, the IPv6 SIP user agent could take advantage of knowing that its own IPv4 address is public and deduct which peer address to use without using STUN. If the answer contains an address which was derived from an alternative in the offer, that address will have best connectivity. If the answer does not contain any derived address, it means that the peer has a local public IPv4 address, which will be the alternative with highest priority in the answer.

At this point, the IPv6 SIP user agent sends back and answer that only differs from its previous answer (shown in Figure 36) in the version number (o= field).

7. Security Considerations

TODO.

8. IANA Considerations

There are no IANA considerations associated with this specification.

9. Acknowledgements

The authors would like to thank Douglas Otis, Karim El Malki and Francois Audet for their comments and input.

10 Informative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.

[3] Rosenberg, J. and H. Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", RFC 3581, August 2003.

[4] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Nettwork Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP)", draft-rosenberg-sipping-ice-01 (work in progress), July 2003.

[5] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[6] Rosenberg, J., Weinberger, J., Huitema, C. and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[7] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-04 (work in progress), February 2004.

[8] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[9] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[10] Malki, K., "IPv6-IPv4 Translators in 3GPP Networks", draft-elmalki-v6ops-3gpp-translator-00 (work in progress), June 2003.

[11] Camarillo, G. and J. Rosenberg, "The Alternative Semantics for the Session Description Protocol Grouping Framework", draft-camarillo-mmusic-alt-02 (work in progress), October 2003.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.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 procedures with respect to rights in RFC 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.

Network Working Group R. Sparks
Internet-Draft dynamicsoft
Expires: December 31, 2004 July 2, 2004

Multiple Dialog Usages in the Session Initiation Protocol
draft-sparks-sipping-dialogusage-00

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/lid-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 31, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

Several methods in the Session Initiation Protocol can create an association between endpoints known as a dialog. Some of these methods can also create a new association within an existing dialog. These multiple associations, or dialog usages, require carefully coordinated processing as they have independent life-cycles, but share common dialog state.

This memo argues that multiple dialog usages should be avoided. It discusses alternatives to their use and clarifies essential behavior for elements that cannot currently avoid them.

This is an informative document and makes no normative statements of any kind.

Table of Contents

1. Introduction

Several methods in SIP can establish a dialog. When they do so, they also establish an association between the endpoints within that dialog. This assocation has been known for some time as a "dialog usage" in the developer community. A dialog initiated with an INVITE request has an invite usage. A dialog initiated with a SUBSCRIBE request has a subscribe usage.

Dialogs with multiple usages arise from actions like a REFER or SUBSCRIBE issued inside a dialog established with an INVITE request. Multiple REFERs within a dialog create multiple subscriptions, each of which is a new dialog usage sharing common dialog state. This state includes:

- the Call-ID
- the local Tag
- the remote Tag
- the local CSeq
- the remote CSeq
- the Route-set
- the local contact
- the remote target

A dialog comes into existence with the creation of the first usage, and continues to exist until the last usage is terminated (reference counting). Unfortunately, many of the usage management aspects of SIP, such as authentication, were originally designed with the implicit assumption that there was one usage per dialog. The resulting mechanisms have mixed effects, some influencing the usage, and some influencing the entire dialog.

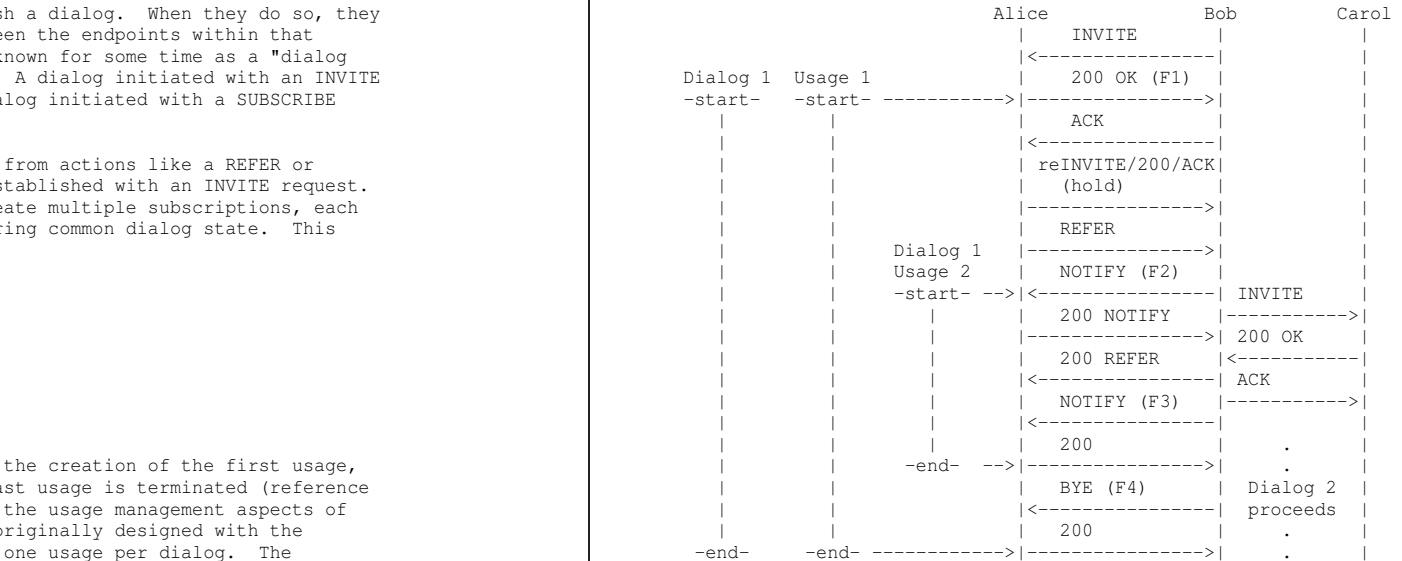
The current specifications define two usages, invite and subscribe. A dialog can share up to one invite usage and arbitrarily many subscribe usages. The pseudo-dialog behavior of REGISTER could be considered a third usage. Fortunately, no existing implementations have attempted to mix a registration usage with any other usage.

2. Examples of Multiple Usages

2.1 Transfer

In Figure 1, Alice transfers a call she received from Bob to Carol. A dialog (and an invite dialog usage) between Alice and Bob came into being with the 200 OK labeled F1. A second usage (a subscription to event refer) springs into being with the NOTIFY labeled F2. This second usage ends when the subscription is terminated by the NOTIFY transaction labeled F3. The dialog still has one usage (the invite usage), which lasts until the BYE transaction labeled F4. At this

point, the dialog has no remaining usages, so it ceases to exist.



Message Details (abridged to show only dialog or usage details)

F1

```
SIP/2.0 200 OK
Call-ID: dialog1@bob.example.com
CSeq: 100 INVITE
To: <sip:Alice@alice.example.com>;tag=alicetag1
From: <sip:Bob@bob.example.com>;tag=bobtag1
Contact: <sip:aliceinstance@alice.example.com>
```

F2

```
NOTIFY sip:aliceinstance@alice.example.com SIP/2.0
Event: refer
Call-ID: dialog1@bob.example.com
CSeq: 101 NOTIFY
To: <sip:Alice@alice.example.com>;tag=alicetag1
From: <sip:Bob@bob.example.com>;tag=bobtag1
Contact: <sip:bobinstance@bob.example.com>
```

F3

```
NOTIFY sip:aliceinstance@alice.example.com SIP/2.0
Event: refer
Subscription-State: terminated;reason=noresource
Call-ID: dialog1@bob.example.com
CSeq: 102 NOTIFY
To: <sip:Alice@alice.example.com>;tag=alicetag1
From: <sip:Bob@bob.example.com>;tag=bobtag1
Contact: <sip:bobinstance@bob.example.com>
Content-Type: message/sipfrag

SIP/2.0 200 OK
```

F4

```
BYE sip:aliceinstance@alice.example.com SIP/2.0
Call-ID: dialog1@bob.example.com
CSeq: 103 BYE
To: <sip:Alice@alice.example.com>;tag=alicetag1
From: <sip:Bob@bob.example.com>;tag=bobtag1
Contact: <sip:bobinstance@bob.example.com>
```

Figure 1

2.2 Reciprocal Subscription

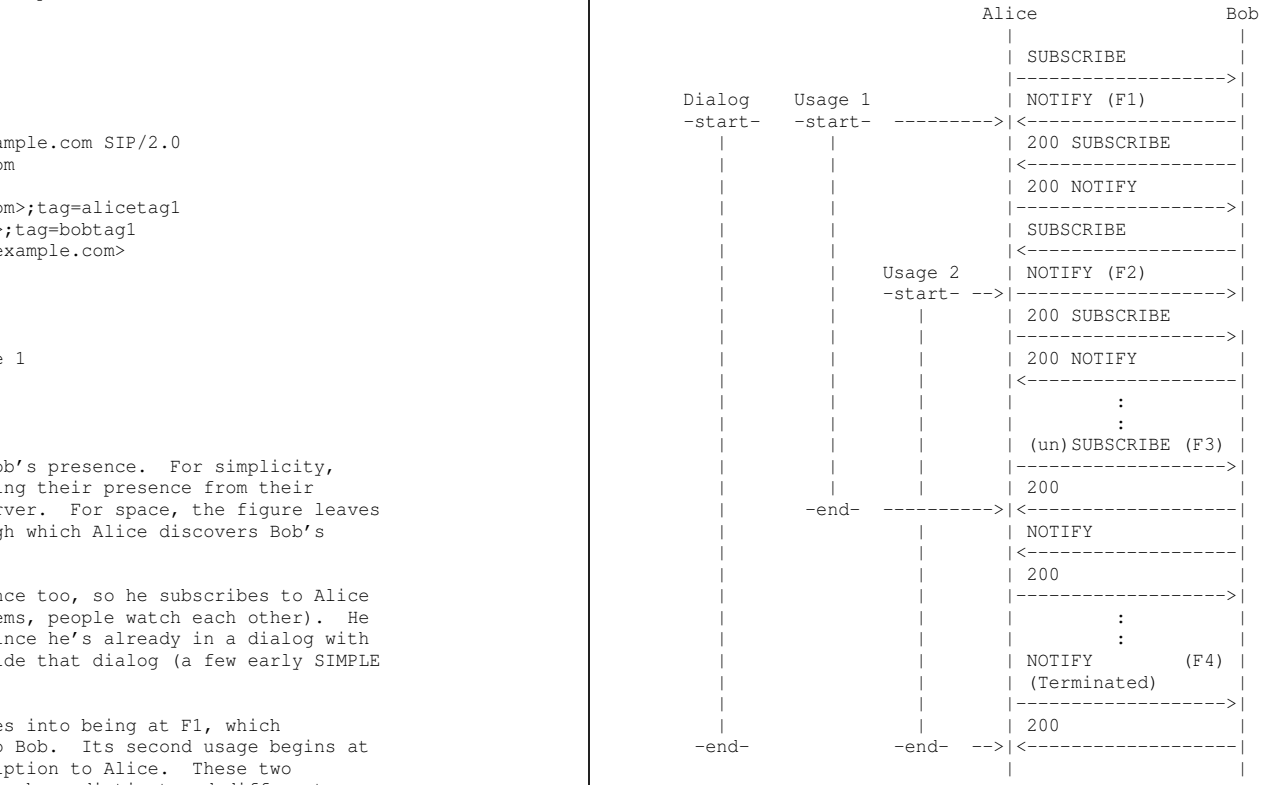
In Figure 2, Alice subscribes to Bob's presence. For simplicity, assume Bob and Alice are both serving their presence from their endpoints instead of a presence server. For space, the figure leaves out any rendezvous signaling through which Alice discovers Bob's endpoint.

Bob is interested in Alice's presence too, so he subscribes to Alice (in most deployed presence/IM systems, people watch each other). He decides skip the rendezvous step since he's already in a dialog with Alice, and sends his SUBSCRIBE inside that dialog (a few early SIMPLE clients behaved exactly this way).

The dialog and its first usage comes into being at F1, which establishes Alice's subscription to Bob. Its second usage begins at F2, which establishes Bob's subscription to Alice. These two subscriptions are independent - they have distinct and different expirations, but they share all the dialog state.

The first usage ends when Alice decides to unsubscribe at F3. Bob's

subscription to Alice, and thus the dialog, continues to exist. Alice's UA must maintain this dialog state even though the subscription that caused it to exist in the first place is now over. The second usage ends when Alice decides to terminate Bob's subscription at F4 (she's probably going to reject any attempt on Bob's part to resubscribe until she's ready to subscribe to Bob again). Since this was the last usage, the dialog also terminates.



Message Details (abridged to show only dialog or usage details)

F1

```
NOTIFY sip:aliceinstance@alice.example.com SIP/2.0
```

```
    Event: presence
    Subscription-State: active;expires=600
    Call-ID: alicecallid1@alice.example.com
    From: <sip:Bob@bob.example.com>;tag=bobtag2
    To: <sip:Alice@alice.example.com>;tag=alicetag2
    CSeq: 100 NOTIFY
    Contact: <sip:bobinstance@bob.example.com>

F2
    NOTIFY sip:bobinstance@bob.example.com SIP/2.0
    Event: presence
    Subscription-State: active;expires=1200
    Call-ID: alicecallid1@alice.example.com
    To: <sip:Bob@bob.example.com>;tag=bobtag2
    From: <sip:Alice@alice.example.com>;tag=alicetag2
    CSeq: 500 NOTIFY
    Contact: <sip:aliceinstance@alice.example.com>

F3
    SUBSCRIBE sip:bobinstance@bob.example.com SIP/2.0
    Event: presence
    Expires: 0
    Call-ID: alicecallid1@alice.example.com
    To: <sip:Bob@bob.example.com>;tag=bobtag2
    From: <sip:Alice@alice.example.com>;tag=alicetag2
    CSeq: 501 SUBSCRIBE
    Contact: <sip:aliceinstance@alice.example.com>

F4
    NOTIFY sip:bobinstance@bob.example.com SIP/2.0
    Event: presence
    Subscription-State: terminated;reason=deactivated
    Call-ID: alicecallid1@alice.example.com
    To: <sip:Bob@bob.example.com>;tag=bobtag2
    From: <sip:Alice@alice.example.com>;tag=alicetag2
    CSeq: 502 NOTIFY
    Contact: <sip:aliceinstance@alice.example.com>
```

Figure 2

3. Proper Handling of Multiple Usages

The examples in Section 2 show straightforward cases where it is fairly obvious when the dialog begins and ends. Unfortunately, there are many scenarios where such clarity is not present. For instance,

in Figure 1, what would it mean if the response to the NOTIFY (F2) were a 481? Does that simply terminate the refer subscription, or does it destroy the entire dialog? This section explores the problem spots with multiple usages that have been identified to date.

3.1 A survey of the effect of failure responses on usages and dialogs

For this survey, consider a subscribe usage inside a dialog established with an invite usage. Unless stated otherwise, we'll discuss the effect on each usage and the dialog when a client issuing a NOTIFY inside the subscribe usage receives a failure response (such as a transferee issuing a NOTIFY to event refer).

This survey is written from the perspective of a client receiving the error response. The effect on dialogs and usages at the server issuing the response is the same.

3xx responses: Redirection mid-dialog is not well understood in SIP, but whatever effect it has impacts the entire dialog and all of its usages equally. In our example scenario, both the subscription and the invite usage would be redirected by this single response.

400 and unrecognized 4xx responses: These responses affect only the NOTIFY transaction, not the subscription, the dialog it resides in (beyond affecting the local CSeq), or any other usage of that dialog. In general, the response is a complaint about this transaction, not the usage or dialog the transaction occurs in.

401 Unauthorized ,407 Proxy Authentication Required: This request, not the subscription or dialog, is being challenged. The usages and dialog are not terminated.

402 Payment Required: This is a reserved response code. If encountered, it should be treated as an unrecognized 4xx.

403 Forbidden: This response terminates the subscription, but has no effect on any other usages of the dialog. In our example scenario, the invite usage continues to exist. Similarly, if the 403 came in response to a reINVITE, the invite usage would be terminated, but not the subscription.

404 Not Found: This response destroys the dialog and all usages sharing it. The Request-URI that is being 404ed is the remote target set by the Contact provided by the peer. Getting this response means something has gone fundamentally wrong with the dialog state.

405 Method Not Allowed: In our example scenario, this response destroys the subscription, but not the invite usage or the dialog. It's an aberrant case for NOTIFYs to receive a 405 since they only come as a result to something that creates subscription. In general, a 405 within a given usage affects only that usage, but does not affect other usages of the dialog.

406 Not Acceptable: These responses concern details of the message in the transaction. Subsequent requests in this same usage may succeed. Neither the usage nor dialog is terminated, other usages sharing this dialog are unaffected.

408 Request Timeout: OPEN ISSUE. 3261 explicitly says this terminates dialogs. Does it really mean it terminates the invite usage? Should it truly tear down all usages on a dialog where it occurs? More on this in Section 3.2.

410 Gone: This response destroys the dialog and all usages sharing it. The Request-URI that is being rejected is the remote target set by the Contact provided by the peer. Similar to 404, getting this response means something has gone fundamentally wrong with the dialog state, its slightly less aberrant in that the other endpoint recognizes that this was once a valid URI that it isn't willing to respond to anymore.

412 Conditional Request Failed:
413 Request Entity Too Large:
414 Request-URI Too Long:
415 Unsupported Media Type: These responses concern details of the message in the transaction. Subsequent requests in this same usage may succeed. Neither the usage nor dialog is terminated, other usages sharing this dialog are unaffected.

416 Unsupported URI Scheme: Similar to 404 and 410, this response came to a request whose Request-URI was provided by the peer in a Contact header field. Something has gone fundamentally wrong, and the dialog and all of its usages are destroyed.

420 Bad Extension, 421 Extension Required: These responses are objecting to the request, not the usage. The usage is not affected. The dialog is only affected by a change in its local CSeq. No other usages of the dialog are affected.

423 Interval Too Brief: This response won't happen in our example scenario, but if it came in response to a reSUBSCRIBE, the subscribe usage is not destroyed (or otherwise affected). No other usages of the dialog are affected.

429 Provide Referrer Identity: This response won't be returned to a NOTIFY as in our example scenario, but when it is returned to a REFER, it is objecting to the REFER request itself, not any usage the REFER occurs within. The usage is unaffected. Any other usages sharing this dialog are unaffected. The dialog is only affected by a change in its local CSeq.

480 Temporarily Unavailable: OPEN ISSUE: Similar to 404,410 this response is to a R-URI that was provided by the peer in a Contact. Is it reasonable for a request to such a URI to return a 480? For instance, if someone places a call on hold and activates Do-not-disturb, would 480 be a reasonable response to decline reINVITEs? We need more clarity around what a mid-usage 480 means. I propose we declare it an error and that this section has an answer like 404s.

481 Call/Transaction Does Not Exist: This response indicates that the peer has lost its copy of the dialog state. The dialog and any usages sharing it are destroyed.

482 Loop Detected: This response is aberrant mid-dialog. It will only occur if the Record-Route header field was improperly constructed by the proxies involved in setting up the dialog's initial usage, or if a mid-dialog request forks and merges (which should never happen). Future requests using this dialog state will also fail. The dialog and any usages sharing it are destroyed. OPEN ISSUE: This response may have been triggered by method (and perhaps usage) specific handling. Is destroying the entire dialog too severe?

483 Too Many Hops: Similar to 482, receiving this mid-dialog is aberrant. Unlike 482, recovery may be possible by increasing Max-Forwards (assuming that the requester did something strange like using a smaller value for Max-Forwards in mid-dialog requests than it used for an initial request). If the request isn't tried with an increased Max-Forwards, then the agent should attempt to gracefully terminate this usage and all other usages that share its dialog. OPEN ISSUE: Is this the right behavior, or should we just declare the dialog terminated?

484 Address Incomplete, 485 Ambiguous: Similar to 404 and 410, these responses came to a request whose Request-URI was provided by the peer in a Contact header field. Something has gone fundamentally wrong, and the dialog and all of its usages are destroyed.

486 Busy Here: This response is non-sensical in our example scenario, or in any scenario where this response comes inside an established usage. If it occurs in that context, it should be treated as an unknown 4xx response. The usage, and any other usages sharing its dialog are unaffected. The dialog is only affected by the change in its local CSeq. If this response is to a request that is attempting to establish a new usage within an existing dialog (such as an INVITE sent within a dialog established by a subscription), the request fails, no new usage is created, and no other usages are affected.

487 Request Terminated: This response speaks to the disposition of a particular request (transaction). The usage in which that request occurs is not affected by this response (it may be affected by another associated request within that usage). No other usages sharing this dialog are affected.

488 Not Acceptable Here: This response is objecting to the request, not the usage. The usage is not affected. The dialog is only affected by a change in its local CSeq. No other usages of the dialog are affected.

489 Bad Event: In our example scenario, [3] declares that the subscription usage in which the NOTIFY is sent is terminated. The invite usage is unaffected and the dialog continues to exist. This response is only valid in the context of SUBSCRIBE and NOTIFY. UAC behavior for receiving this response to other methods is not specified, but treating it as an unknown 4xx is a reasonable practice.

491 Request Pending: This response addresses in-dialog request glare. Its affect is scoped to the request. The usage in which the request occurs is not affected. The dialog is only affected by the change in its local CSeq. No other usages sharing this dialog are affected.

493 Undecipherable: This response objects to the request, not the usage. The usage is not affected. The dialog is only affected by a change in its local CSeq. No other usages of the dialog are affected.

494 Security Agreement Required: This response is objecting to the request, not the usage. The usage is not affected. The dialog is only affected by a change in its local CSeq. No other usages of the dialog are affected.

500 and 5xx unrecognized responses: These responses are complaints against the request (transaction), not the usage. If the response contains a Retry-After header field value, the server thinks the condition is temporary and the request can be retried after the indicated interval. This usage, and any other usages sharing the dialog are unaffected. If the response does not contain a Retry-After header field value, the UA may decide to retry after an interval of its choosing or attempt to gracefully terminate the usage. Whether or not to terminate other usages depends on the application. If the UA receives a 500 (or unrecognized 5xx) in response to an attempt to gracefully terminate this usage, it can treat this usage as terminated. If this is the last usage sharing the dialog, the dialog is also terminated.

501 Not Implemented: This would be a degenerate response in our example scenario since the NOTIFY is being sent as part of an established subscribe usage. In this case, the UA knows the condition is unrecoverable and should stop attempting to send NOTIFYs on this usage. (It may or may not destroy the usage. If it remembers the bad behavior, it can reject any refresh subscription). In general, this response may or may not affect the usage (a 501 to an unknown method or an INFO will not end an invite usage). It will never affect other usages sharing this usage's dialog.

502 Bad Gateway: OPEN ISSUE: I think this is similar to "Loop Detected".

503 Service Unavailable: As per [2], the logic handling locating SIP servers for transactions may handle 503 requests (effectively sequentially forking at the endpoint based on DNS results). If this process does not yield a better response, a 503 may be returned to the transaction user. Like a 500 response, the error is a complaint about this transaction, not the usage. Because this response occurred in the context of an established usage (hence an existing dialog), the route-set has already been formed and any opportunity to try alternate servers (as recommended in [1] has been exhausted by the RFC3263 logic. The response should be handled as described for 500 earlier in this memo.

504 Server Time-out: It is not obvious under what circumstances this response would be returned to a request in an existing dialog. If it occurs it should have the same affect on the dialog and its usages as described for unknown 5xx responses.

505 Version Not Supported, 513 Message Too Large: These responses are objecting to the request, not the usage. The usage is not affected. The dialog is only affected by a change in its local CSeq. No other usages of the dialog are affected.

580 Precondition Failure: This response is objecting to the request, not the usage. The usage is not affected. The dialog is only affected by a change in its local CSeq. No other usages of the dialog are affected.

600 and 6xx unrecognized responses: Unlike 400 Bad Request, a 600 response code says something about the recipient user, not the request that was made. This end user is stating an unwillingness to communicate. OPEN ISSUE: It is not clear what this means in response to a mid-dialog request. I propose this behavior if you receive a 600 or unknown 6xx: If the response contains a Retry-After header field value, the user is indicating willingness to communicate later and the request can be retried after the indicated interval. This usage, and any other usages sharing the dialog are unaffected. If the response does not contain a Retry-After header field value, the UA may decide to retry after an interval of its choosing or attempt to gracefully terminate the usage. Whether or not to terminate other usages depends on the application. If the UA receives a 600 (or unrecognized 6xx) in response to an attempt to gracefully terminate this usage, it can treat this usage as terminated. If this is the last usage sharing the dialog, the dialog is also terminated.

603 Decline: This response declines the action indicated by the associated request. It can be used, for example, to decline a hold or transfer attempt. Receiving this response does NOT terminate the usage it occurs in. Other usages sharing the dialog are unaffected.

604 Does Not Exist Anywhere: Like 404, this response destroys the dialog and all usages sharing it. The Request-URI that is being 604ed is the remote target set by the Contact provided by the peer. Getting this response means something has gone fundamentally wrong with the dialog state.

606 Not Acceptable: This response is objecting to aspects of the associated request, not the usage the request appears in. The usage is unaffected. Any other usages sharing the dialog are unaffected. The only affect on the dialog is the change in the local CSeq.

3.2 Transaction timeouts

[1] states that a UAC should terminate a dialog (by sending a BYE) if no response is received for a request sent within a dialog. This recommendation should have been limited to the invite usage instead of the whole dialog. [3] states that a timeout for a NOTIFY removes a subscription, but a SUBSCRIBE that fails with anything other than a 481 does not. Given these statements, it is unclear whether a refresh SUBSCRIBE issued in a dialog shared with an invite usage destroys either usage or the dialog if it times out.

Generally, a transaction timeout should affect only the usage in which the transaction occurred. Other uses sharing the dialog should not be affected. In the worst case of timeout due to total transport failure, it may require multiple failed messages to remove all usages from a dialog (at least one per usage).

There are some mid-dialog messages that never belong to any usage. If they timeout, they will have no effect on the dialog or its usages.

3.3 Matching requests to usages

For many mid-dialog requests, identifying the usage they belong to is obvious. A dialog can have at most one invite usage, so any INVITE, UPDATE, PRACK, ACK, CANCEL, BYE, or INFO requests belong to it. The usage (i.e. the particular subscription) SUBSCRIBE, NOTIFY, and REFER requests belong to can be determined from the Event header field of the request. REGISTER requests within a (pseudo)-dialog belong to the registration usage. (As mentioned before, implementations aren't mixing registration usages with other usages, so this document isn't exploring the consequences of that bad behavior).

According to [1], "an OPTIONS request received within a dialog generates a 200 OK response that is identical to one constructed outside a dialog and does not have any impact on that dialog". Thus OPTIONS does not belong to any usage. Only those failures discussed in Section 3.1 and Section 3.2 that destroy entire dialogs will have any effect on the usages sharing the dialog with a failed OPTIONS request.

MESSAGE requests are not currently allowed inside a dialog (though some implementations use it that way, against the standard recommendation). As it is not meant to be part of any given dialog, it cannot be part of any given usage. A failed MESSAGE request should have similar effects on a dialog and its usages as a failed OPTIONS request.

Mid-dialog requests with unknown methods cannot be matched with a usage. Servers will return a failure response (likely a 501). The effect on the dialog and its usages at either the client or the server should be similar to that of a failed OPTIONS request.

3.4 Target refresh requests

Target refresh requests update the remote target of a dialog when they are successfully processed. The currently defined target refresh requests are INVITE, UPDATE, SUBSCRIBE and NOTIFY (clarified in a bug against RFC3565). REFER could also be a target refresh request since it can establish a new usage (and even a new dialog). (OPEN ISSUE: Is REFER a target refresh request?)

The remote target is part of the dialog state. When a target refresh request affects it, it affects it for ALL usages sharing that dialog. If a subscription and invite usage are sharing a dialog, sending a refresh SUBSCRIBE with a different contact will cause reINVITEs from the peer to go to that different contact.

A UAS will only update the remote target if it sends a 200 class response to a target refresh request. A UAC will only update the remote target if it receives a 200 class response to a target refresh request. Again, any update to a dialog's remote target affects all usages of that dialog.

3.5 Refreshing and Terminating Usages

Subscription and registration usages expire over time and must be refreshed (with a refresh SUBSCRIBE for example). This expiration is usage state, not dialog state. If several subscriptions share a dialog, refreshing one of them has no effect on the expiration of the others.

Normal termination of a usage has no effect on other usages sharing the same dialog. For instance terminating a subscription with a NOTIFY/Subscription-State: terminated will not terminate an invite usage sharing its dialog. Likewise, ending an invite usage with a BYE does not terminate any active Event: refer subscriptions established on that dialog.

Abnormal termination can effect all usages on a dialog. Rejecting a NOTIFY with a 481 (incorrectly recommended in the past as an inexpensive way to terminate a REFER subscription) destroys the dialog and all of its usages.

3.6 Refusing new usages

As the survey of the effect of failure responses shows, care must be taken when refusing a new usage inside an existing dialog. Choosing the wrong response code will terminate the dialog and all of its usages. Generally, returning a 603 Decline is the safest way to refuse a new usage.

3.7 Replacing usages

[5] defines a mechanism through which one usage can replace another. It can be used, for example, to associate the two dialogs a transfer target is involved in during an attended transfer. It is written using the term "dialog", but its intent was to only affect the invite usage of the dialog it targets. Any other usages inside that dialog are unaffected. For some applications, the other usages may no longer make sense, and the application may terminate them as well.

However, the interactions between Replaces and multiple dialog usages have not been well explored. More discussion of this topic is needed. Implementers should avoid this scenario completely.

4. Avoiding Multiple Usages

Processing multiple usages correctly is not completely understood. What is understood is difficult to implement and is very likely to lead to interoperability problems. The best way to avoid the trouble that comes with such complexity is to avoid it altogether.

When designing new applications that use SIP dialogs, do not construct multiple usages. If a peer attempts to create a second usage inside a dialog, refuse it.

Unfortunately, there are existing applications, like transfer, that currently entail multiple usages, so the simple solution of "don't do it" will require some transitional work. This section will look at the pressures that led to these existing multiple usages and suggest alternatives.

When executing a transfer, the transferor and transferee currently share an invite usage and a subscription usage within the dialog between them. This is a result of sending the REFER request within the dialog established by the invite usage. Implementations were led to this behavior by two primary pressures:

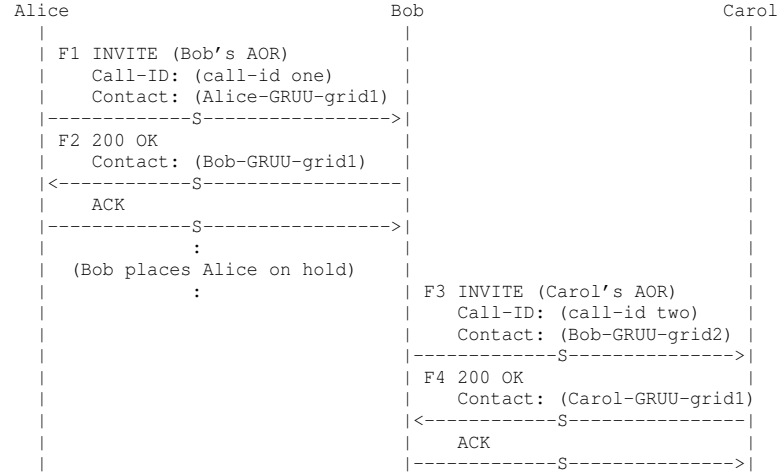
1. There was no way to ensure that a REFER on a new dialog would reach the particular endpoint involved in a transfer. Many factors, including details of implementations and changes in proxy routing between an INVITE and a REFER could cause the REFER

to be sent to the wrong place. Sending the REFER down the existing dialog ensured it got to the endpoint we were already talking to.

2. It was unclear how to associate an existing invite usage with a REFER arriving on a new dialog, where it was completely obvious what the association was when the REFER came on the invite usage's dialog.
3. There were concerns with authorizing out-of-dialog REFERs. The authorization policy for REFER in most implementations piggybacks on the authorization policy for INVITE (which is, in most cases, based simply on "I placed or answered this call").

GRUUs ([6]) have been defined specifically to address problem 1. Problem 2 can be addressed using a GRUU's grid parameter. In the immediate term, this solution to problem 2 allows the existing REFER authorization policy to be reused. Figure 3 shows a transfer where any given dialog has exactly one usage.

Each message in this flow passes through a server at example.com, which forwards messages to the endpoints based on the AOR or GRUU in the Request-URI. This hop through the server has been removed from the diagram to make it easier to read. An "S" appears in the middle of each arrow as a reminder of the visit to this intermediary.



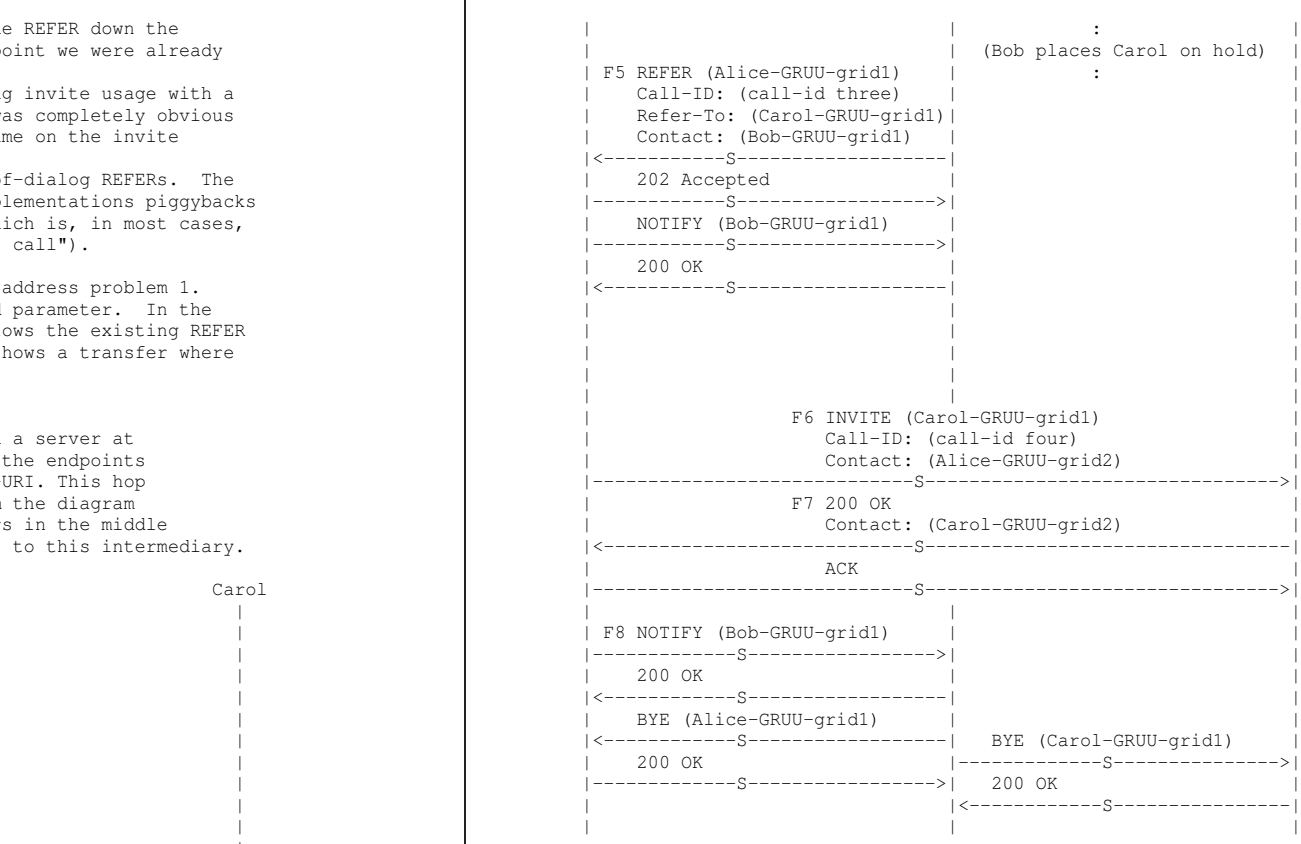


Figure 3: Transfer without dialog reuse

In message F1, Alice invites Bob indicating support for GRUUs (and offering a GRUU for herself):

Message F1 (abridged, detailing pertinent fields)

```
INVITE sip:bob@example.com SIP/2.0
Call-ID: 13jfdwer230jsdw@alice.example.com
Supported: gruu
Contact: <sip:aanewmr203raswdf@example.com;grid=au9a3e>
```

Message F2 lets Alice know that Bob understands GRUUs. If Bob did not indicate this support, the original multi-usage approach to transfer would have to be used.

Message F2 (abridged, detailing pertinent fields)

```
SIP/2.0 200 OK
Supported: gruu
Contact: <sip:boaiidfjjereis@example.com;grid=baierac>
```

Bob decides to try to transfer Alice to Carol, so he puts Alice on hold and sends message F3 to Carol. Notice that Bob has provided a different grid to Carol than he provided to Alice. This is a significant part of the solution to problem 2 - if Alice or Carol were to beat Bob to a REFER, this will let Bob know which invite usage (the one with Alice or the one with Carol) to affect.

Message F3 (abridged, detailing pertinent fields)

```
INVITE sip:carol@example.com SIP/2.0
Call-ID: 23rasdnfoa39i4jnasdf@bob.example.com
Supported: gruu
Contact: <sip:boaiidfjjereis@example.com;grid=bc923a3d>
```

Carol indicates her own support of GRUU and provides her GRUU for this dialog in message F4:

Message F4 (abridged, detailing pertinent fields)

```
SIP/2.0 200 OK
Supported: gruu
Call-ID: 23rasdnfoa39i4jnasdf@bob.example.com
To: <sip:carol@example.com>;tag=foiew3n
From: <sip:bob@example.com>;tag=baeih23n
Contact: <sip:c239fniuweorw9sdfn@example.com;grid=cbfnei2>
```

After consulting Carol, Bob places her on hold and refers Alice to

her using message F5. Notice that the Refer-To URI is Carol's GRUU, and that this is on a different Call-ID than message F1. (The URI in the Refer-To header is line-broken for readability in this draft, it would not be valid to break the URI this way in a real message)

Message F5 (abridged, detailing pertinent fields)

```
REFER sip:aanewmr203raswdf@example.com;grid=au9a3e SIP/2.0
Call-ID: 39fa99r0329493asdsf3n@bob.example.com
Refer-To: <sip:c239fniuweorw9sdfn@example.com;grid=cbfnei2
           ?Replaces=23rasdnfoa39i4jnasdf@bob.example.com;
            to-tag=foiew3n;from-tag=baeih23n>
Supported: gruu
Contact: <sip:boaiidfjjereis@example.com;grid=baierac>
```

Alice accepts this REFER, sends Bob the obligatory immediate NOTIFY, and proceeds to INVITE Carol with message F6. Notice that Alice gives Carol a GRUU with a different grid than she gave Bob. If Bob decided not to terminate his dialog with Alice (possibly sending her another REFER) and/or Carol decided to transfer Alice again, this becomes an important part of associating Bob or Carol's REFER with the correct invite usage.

Message F6 (abridged, detailing pertinent fields)

```
INVITE sip:c239fniuweorw9sdfn@example.com;grid=cbfnei2 SIP/2.0
Call-ID: 4zsd9f234jasdfasn3jsad@alice.example.com
Replaces: 23rasdnfoa39i4jnasdf@bob.example.com;
          to-tag=foiew3n;from-tag=baeih23n
Supported: gruu
Contact: <sip:aanewmr203raswdf@example.com;grid=ac99asn>
```

Carol accepts Alice's invitation to replace her dialog (invite usage) with Bob with F7. For the same reasons listed above, Carol hands Alice a different grid than the one she handed Bob (which was the one Alice used to reach her in F6).

Message F7 (abridged, detailing pertinent fields)

```
SIP/2.0 200 OK
Supported: gruu
Contact: <sip:c239fniuweorw9sdfn@example.com;grid=canasdi>
```

Alice notifies Bob that the REFERenced INVITE succeeded with F8:

Message F8 (abridged, detailing pertinent fields)

```
NOTIFY sip:boaiidfjjereis@example.com;grid=baierac SIP/2.0
Subscription-State: terminated;reason=noresource
Contact: <sip:aanewmr203raswdf@example.com;grid=au9a3e>
Content-Type: message/sipfrag

SIP/2.0 200 OK
```

Bob then ends his invite usages with both Alice and Carol using BYEs.

Generalizing what was said several times during that flow: Using a different grid for each usage between two endpoints lets endpoints involved in more than one dialog figure out which dialog is related to a new out-of-dialog request. Having that association can affect whether this new out-of-dialog request is accepted.

One potential application for REFER that has been discussed at several working group meetings is using an out-of-dialog REFER to ask an endpoint to join a conference. An endpoint receiving such a REFER would have to authorize it (by prompting its user for permission perhaps). If this REFER arrived while the user was in another call, the lack of a grid parameter matching the call that is ongoing lets the UA know that this REFER is not a transfer of the existing call.

5. Conclusion

Handling multiple usages within a single dialog is complex and introduces scenarios where the right thing to do is not clear. Implementations should avoid entering into multiple usages whenever possible. New applications should be designed to never introduce multiple usages.

There are some accepted SIP practices, including transfer, that currently require multiple usages. Recent work, most notably GRUU, makes those practices unnecessary. The standardization of those practices and the implementations should be revised as soon as possible to use only single-usage dialogs.

6 Informative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

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

[4] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[5] Biggs, B., Dean, R. and R. Mahy, "The Session Inititation Protocol (SIP) 'Replaces' Header", draft-ietf-sip-replaces-04 (work in progress), August 2003.

[6] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-01 (work in progress), February 2004.

Author's Address

Robert J. Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

EMail: rsparks@dynamicsoft.com

Appendix A. Acknowledgments

The ideas in this draft have been refined over several IETF meetings with many participants. Significant contribution was provided by Adam Roach, Alan Johnston, Ben Campbell, Cullen Jennings, Jonathan Rosenberg and Rohan Mahy. Members of the reSIProcate project (http://www.sipfoundry.org/reSIProcate) also shared their difficulties and discoveries while implementing multiple-usage dialog handlers.

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 procedures with respect to rights in RFC 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.