Internet Engineering Task Force
Internet Draft

J. Elwell
Siemens
V. Venkataramanan
Sylantro Systems Corp

draft-elwell-sip-state-update-02.txt
Expires: December 2004

June 2004

State update during a SIP dialog

Status of this Memo

Copyright Notice

Abstract

This document examines the need for updating state information, such as remote party identity, during a SIP dialog. It explores existing mechanisms that might be appropriate and proposes minor clarifications to existing RFCs and drafts to achieve this.

Table of Contents

1 Introduction

Certain information exchanged between SIP [1] UAs during an INVITE transaction can be stored at a receiving UA for the lifetime of the resulting dialog and/or made available to the user (e.g., via a display in the case of a human user). One example is the identity of the peer user (as supplied in the To or From header, the Authenticated Identity Body (AIB) or the P-Asserted-Identity header). Another example is the Subject header. This information forms part of the state of a dialog at a UA.

Under certain circumstances some of this state information may need to be changed. For example, when interworking with a PSTN, there may be a change of party in the PSTN (e.g., because of call transfer). The PSTN party identity sent to the peer SIP UA in the To or From header, the AIB or the P-Asserted-Identity header during dialog establishment normally reflects the identity of the PSTN user. If the identity of the PSTN user changes during the lifetime of the dialog, this information needs to be updated at the UA.

A second example involves 3rd party call control. A B2BUA performing 3rd party call control can perform actions such as call transfer that cause a change of membership of the call. An existing UA that remains involved in the call and retains its dialog with the B2BUA needs to be updated with the identity of the new remote party.

A third example also involves 3rd party call control. In this case a B2BUA forms a conference and acts as a conference focus. It therefore needs to indicate this to any existing UA whose dialog is "transferred" into that conference.

A fourth example is where a user changes the subject during a dialog. The revised subject needs to be communicated to the remote UA.

Existing mechanisms for changing a session during a dialog (re-INVITE and UPDATE transactions) may be a suitable basis for making other state changes, but it is not at present clear if and how these mechanisms are applicable.

2 State information subject to change during a dialog

The following information exchanged during the INVITE transaction can change during the lifetime of the resulting dialog.

- Call-Info header.

- Alert-Info header (early dialogs only).

- Contact header (change of feature tags [3]).

- Reply-To header.

- Subject header.

- P-Asserted-Identity header [4].

- Privacy header [5] (for use in conjunction with updated identity information).

- Authenticated Identity Body (AIB) [6].

- Bodies with Content-Disposition "icon" or "alert".

In addition consideration should be given to the following headers: Allowed, Supported, Required. These would not normally be expected to change at a UA reporting a state change. However, there might be some B2BUA arrangements where Allowed, Supported and Required are sent by the B2BUA on behalf of another UA, and if that other UA changes, these headers might change.

Current studies on including location information in SIP messages [7] should also take into account the need to update location information during a dialog. One application might be a direction service, which provides directions from the caller's current location to a given venue. As the caller's location changes (e.g., as detected by GPS), the location information may need to be updated. Another application is emergency calling, where more accurate location information becomes known during the dialog.

3 Applicability of existing mechanisms

3.1 UPDATE

The UPDATE mechanism [2] provides a means for updating the session using a 2-message sequence (request/response) during an INVITE-initiated dialog. Although one of the prime motivations for UPDATE is use during an early dialog (in conjunction with the PRACK method), where re-INVITE cannot be used, UPDATE can also be used during a confirmed dialog. The RFCs concerned are unclear on the use of UPDATE for updating the following state information:

Call-Info header. According to [2], this is optional in an UPDATE request, but no semantics are given. It is unclear whether this is allowed to differ from what was in the INVITE request or response, and if so the meaning of this.

Alert-Info header. According to [2], this is not applicable in an UPDATE request. However, a possible use during an early dialog would be to change the alerting indication or ringback tone.

Contact header. According to [2], this is mandatory in an UPDATE request. In [3] there is mention of updating feature parameters during a dialog.

Reply-To header. This is not allowed in an UPDATE request.

Subject header. According to [2], this is not allowed in an UPDATE request.

P-Asserted-Identity header. According to [4], this header is not allowed in an UPDATE request.

Privacy header. There is no mention in [5] of using this header in an UPDATE request.

AIB. [6] does provide for the use of AIB in a request within the context of an existing dialog. However, it does not mention UPDATE specifically. Also, it does not mention semantics, e.g., whether an AIB in a request in an existing dialog overrides any AIB in the original request or response.

Content-Disposition "icon" or "alert". There is no mention of bodies with these Content-Disposition values, but presumably nothing to prevent their inclusion.

3.2 re-INVITE

The re-INVITE mechanism [1] provides a means for updating the session during an INVITE-initiated dialog. It differs from UPDATE in that it uses a three message sequence (request, response, ACK), and this takes account of possible delays while a user is consulted on the proposed update. Therefore for updating the session on a confirmed dialog, re-INVITE will often be preferred to UPDATE. If state information needs to be updated at the same time as the session, re-INVITE might be the preferred choice. At other times UPDATE might be the preferred choice for updating state information. Another consideration is that some SIP implementations do not currently support UPDATE.

The RFCs concerned are unclear on the use of re-INVITE for updating the following state information:

Call-Info header. According to [1], this is optional in an INVITE request, but there is no specific mention of re-INVITE. It is unclear

whether this is allowed to differ from what was in the original INVITE request or response, and if so the meaning of this.

Contact header. According to [1], this is mandatory in a re-INVITE request. In [3] there is mention of updating feature parameters during a dialog.

Reply-To header. According to [1], this is optional in an INVITE request, but there is no specific mention of re-INVITE. It is unclear whether this is allowed to differ from what was in the original INVITE request or response, and if so the meaning of this.

Subject header. According to [2], this is optional in an INVITE request, but there is no specific mention of re-INVITE. It is unclear whether this is allowed to differ from what was in the original INVITE request or response, and if so the meaning of this.

P-Asserted-Identity header. There is no mention in [4] of using this header in a re-INVITE request.

Privacy header. There is no mention in [5] of using this header in a re-INVITE request.

AIB. [6] does provide for the use of AIB in a request within the context of an existing dialog. However, it does not mention re-INVITE specifically. Also, it does not mention semantics, e.g., whether an AIB in a request in an existing dialog overrides any AIB in the original request or response.

Content-Disposition "icon" or "alert". There is no mention of bodies with these Content-Disposition values in a re-INVITE request, but presumably nothing to prevent their inclusion.

3.3 INVITE with Replaces header

Rather than updating state information for the existing dialog, a new dialog could be created using an INVITE addressed to the remote contact (assuming this is a GRUU) and the Replaces header, thereby causing the new dialog to replace the existing dialog. This is a heavyweight method of achieving the desired results. In particular it requires support of the Replaces header, support of GRUUs, and re-negotiation of the session.

4 Proposal

It is proposed that the use of the UPDATE method for updating state information during a confirmed or early dialog be endorsed. It is also proposed that the use of the re-INVITE method be endorsed for updating state information during a confirmed dialog for cases where

the peer UA does not support UPDATE or when the session is to be updated at the same time.

In order to achieve this, clarifications are required to some existing RFCs:

- RFC 3261: Clarify that a re-INVITE can be used to update dialog-related information during a dialog, i.e., not just a target refresh or a session description change.

- RFC 3311: Clarify that an UPDATE can be used without a session description in order to update dialog-related information. This is already implicit in some places, but a few places seem to require an UPDATE request to contain an SDP offer.

- RFC 3323: This does not specify specific procedures for particular methods and does not make any distinction between a request outside of an existing dialog or a request within a dialog. Therefore as it stands it is in theory equally applicable to UPDATE and re-INVITE as it is to INVITE. However, certain privacy services may not be possible to provide during a dialog if not already provided from the start of the dialog. Its main usefulness seems to be to indicate privacy for an updated P-Asserted-Identity header or an updated Authenticated Identity Body. No changes are proposed.

- RFC 3325: Allow the use of the P-Asserted-Identity header with the UPDATE method. The RFC does not specify specific procedures for particular methods and does not make any distinction between a request outside of an existing dialog or a request within a dialog. Therefore as it stands it is equally applicable to UPDATE (with the proposed change) and re-INVITE as it is to INVITE. Therefore no procedural changes need to be made.

- draft-ietf-sip-authid-body-02: Add mention of use of AIB in re-INVITE or UPDATE.

The remainder of this draft proposes detailed clarifications.

5 Clarifications to RFC 3261[1]

Add the following paragraph to the end of 12 (prior to 12.1):

> "A dialog can also have certain other state information that can change during the lifetime of the dialog. This includes the session description (as negotiated by SDP offer/answer), information from certain other headers and information from certain other bodies. Examples from this RFC include the Alert-Info, Call-Info, Reply-To and Subject headers and bodies with a value of "icon" or "alert" in the Content-Disposition header. Extensions may define other headers

or bodies containing information that contributes to the state of a dialog and can change during the lifetime of the dialog."

Add two new paragraphs to the end of 12.2 (prior to 12.2.1):

"A request, whether or not it is a target refresh request, MAY update certain other information transmitted at the time of dialog establishment. Examples include the following headers from this RFC: Alert-Info, Call-Info, Reply-To and Subject. Other extensions may define other headers that are appropriate for updating during a dialog. In addition, a request, whether or not it is a target refresh request, MAY update a body with a value of "icon" or "alert" in the Content-Disposition header.

"In this RFC, the only request defined that is suitable for updating information during a dialog is re-INVITE (see section 14). Other extensions may define different requests suitable for updating information during a dialog."

Add to the end of the first paragraph of 14.1:

"A UAC MAY send a session description that is unchanged, if the purpose of the re-INVITE is solely for updating dialog-related information."

6 Clarifications to RFC 3311 [2]

General
Call-Info
Alert-Info
Reply-To
Subject

Add to the end of the last paragraph of section 1:

"It can also be sent by a UA within a dialog (early or confirmed) to update dialog-related information in addition to or instead of updating session parameters."

Replace the last sentence of section 3:

Old text: "There are additional constraints on when UPDATE can be used, based on the restrictions of the offer/answer model."

New text: "The UPDATE method can also be used without SDP offer/answer in order to update only dialog-related information and not the session. There are additional constraints on when UPDATE can be used with SDP offer/answer, based on restrictions of the offer/answer model."

Change "MAY" to "SHOULD" in the 2nd paragraph of section 4:

Old text: "An unreliable provisional response MAY contain an Allow header field listing the UPDATE method ...

New text: "An unreliable provisional response SHOULD contain an Allow header field listing the UPDATE method ...

Extend the last sentence of the 3rd paragraph of section 4:

Old text: "Creation of this dialog is necessary in order to receive UPDATE requests from the callee."

New text: "Creation of this dialog is necessary in order to receive UPDATE requests from the callee and in order to receive an SDP offer in an UPDATE request from the caller."

Add the following paragraph at the end of section 4:

"Although a UAS for an INVITE request must use a reliable provisional response in order to ensure that an early dialog is created before issuing an UPDATE request towards the caller, a UAC for an INVITE request will know that an early dialog is established even if it receives an unreliable provisional response. The UAC for the INVITE request MAY then issue an UPDATE request without an SDP offer towards the callee, subject to having received an appropriate Allow header field."

Replace the 3rd sentence of section 5.1:

Old text: "Although UPDATE can be used on confirmed dialogs, it is RECOMMENDED that a re-INVITE be used instead."

New text: "Although UPDATE can be used on confirmed dialogs, it is RECOMMENDED that a re-INVITE be used instead if there is a need to seek user approval."

Replace the last sentence of the 2nd bullet of section 5.1:

Old text: "Of course, it can’t send an UPDATE if it has not received answers to any other offers it sent in either PRACK or UPDATE, or has not generated answers for any other offers it received in an UPDATE from the callee."

New text: "Of course, it can’t send an UPDATE containing an offer if it has not received answers to any other offers it sent in either PRACK or UPDATE, or has not generated answers for any other offers it received in an UPDATE from the callee."

Add a new paragraph at the end of section 5.2:

"If a UA receives an UPDATE request with no session description for an existing dialog, the UA MUST NOT include a session description in the response. In this case the UPDATE request is for the purpose of target refresh or updating other dialog-related information".

Change the following row in table 1:

Old:	Alert-Info	-
New:	Alert-Info	o

Change the following rows in table 2:

Old:	Reply-To		-
New:	Reply-To		o
Old:	Subject	-	-
New:	Subject		o

7 Clarification to RFC 3325 [4]

In section 9.1, change the additional entry to table 1 of RFC 3261 as follows (to make UPDATE optional):

Old:

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
P-Asserted-Identity		adr	-	o	-	o	o	-

SUB	NOT	REF	INF	UPD	PRA
o	o	o	-	-	-

New:

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
P-Asserted-Identity		adr	-	o	-	o	o	-

SUB	NOT	REF	INF	UPD	PRA
o	o	o	-	o	-

8 Clarification to draft-ietf-sip-authid-body-02 [6]

Add the following to the first paragraph of section 5:

> "An AIB in a request within the context of an existing dialog (e.g., re-INVITE, UPDATE) can be used to replace the corresponding identity transmitted at the start of the dialog."

9 Changes

Draft-02: The following changes have been made compared with draft 01:

- Addition of bodies with Content-Disposition "icon" or "alert" to state information that can be changed during a call. (Sections 2, 3.1, 3.2 and 5).

- Addition to section 5 of proposed text for section 12 of RFC3261 elaborating on the state of a dialog.

- Deletion of the open issue in section 5 concerning the ability to have more than one concurrent re-INVITE request outstanding. There was a feeling that it might get too complex.

- Addition of discussion on location information in section 2.

10 Author's Addresses

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

Venkatesh Venkataramanan
Sylantro Systems Corp
910 E Hamilton Ave,
Campbell, CA 95008
Sylantro inc.
email: Venkatesh.Venkataramanan@sylantro.com

11 Normative References

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

[2] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE method", RFC 3311.

[3]J. Rosenberg, H. Schulzrinne, P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", draft-ietf-sip-callee-caps-03.txt (work in progress).

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

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

[6] J. Peterson. "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-02.txt (work in progress).

[7] J. Polk, B. Rosen. "Requirement for Session Initiation Protocol Location Conveyance", draft-ietf-sipping-location-requirements-00.txt (work in progress).

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgement

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

SIP WG
Internet-Draft
Expires: November 3, 2004

J. Peterson
NeuStar

May 5, 2004

SIP Authenticated Identity Body (AIB) Format
draft-ietf-sip-authid-body-03

Status of this Memo

Copyright Notice

Abstract

RFC3261 introduces the concept of adding an S/MIME body to a SIP request or response in order to provide reference integrity over its headers. This document provides a more specific mechanism to derive integrity and authentication properties from an 'authenticated identity body', a digitally-signed SIP message or message fragment. A standard format for such bodies (known as Authenticated Identity Bodies, or AIBs) is given in this document. Some considerations for the processing of AIBs by recipients of SIP messages with such bodies are also given.

Table of Contents

1. Introduction

Section 23.4 of RFC3261 [1] describes an integrity mechanism that relies on signing tunneled 'message/sip' MIME bodies within SIP requests. The purpose of this mechanism is to replicate the headers of a SIP request within a body carried in that request in order to provide a digital signature over these headers. The signature on this body also provides authentication.

The core requirement that motivates the tunnneled 'message/sip' mechanism is the problem of providing a cryptographically verifiable identity within a SIP request. The baseline SIP protocol allows a user agent to express the identity of its user in any of a number of headers. The primary place for identity information asserted by the sender of a request is the From header. The From header field contains a URI (like 'sip:alice@example.com') and an optional display-name (like "Alice") that identifies the originator of the request. A user may have many identities that are used in different contexts.

Typically, this URI is an address-of-record that can be dereferenced in order to contact the originator of the request; specifically, it is usually the same address-of-record under which a user registers their devices in order to receive incoming requests. This address-of-record is assigned and maintained by the administrator of the SIP service in the domain identified by the host portion of the address-of-record. However, the From field of a request can usually be set arbitrarily by the user of a SIP user agent; the From header of a message provides no internal assurance that the originating user can legitimately claim the given identity. Nevertheless, many SIP user agents will obligingly display the contents of the From field as the identity of the originator of a received request (as a sort of caller identification function), much as email implementations display the From field as the sender's identity.

In order to provide the recipient of a SIP message with greater assurance of the identity of the sender, a cryptographic signature can be provided over the headers of the SIP request, which allows the signer to assert a verifiable identity. Unfortunately, a signature over the From header alone is insufficient because it could be cut-and-pasted into a replay or forwarding attack, and more headers are therefore needed to correlated a signature with a request. RFC3261 therefore recommends copying all of the headers from the request into a signed MIME body; however, SIP messages can also be large, and many of the headers in a SIP message would not be relevant to determining the identity of the sender or assuring reference integrity with the request, and moreover some headers may change in transit for perfectly valid reasons. Thus, this large tunneled 'message/sip'

body will almost necessarily be at variance with the headers in a request when it is received by the UAS, and the burden in on the UAS to determine which header changes were legitimate, and which were security violations. It is therefore desirable to find a happy medium - to provide a way of signing just enough headers that the identity of the sender can be ascertained and correlated with the request. 'message/sipfrag' [4] provides a way for a subset of SIP headers to be included in a MIME body; the Authenticated Identity Body (AIB) format described in Section 2 is based on 'message/sipfrag'.

For reasons of end-to-end privacy, it may also be desirable to encrypt AIBs; procedures for this encryption are given in Section 8.

This document proposes that the AIB format should be used instead of the existing tunneled 'message/sip' mechanism described in RFC3261 23.4 in order to provide the identity of the caller; if integrity over other, unrelated headers is required, then the 'message/sip' mechanism should be used.

2. AIB Format

As a way of sharing authenticated identity among parties in the network, a special type of MIME body format, the Authenticated Identity Body (AIB) format, is defined in this section. AIBs allow a party in a SIP transaction to cryptographically sign the headers that assert the identity of the originator of a message, and provide some other headers necessary for reference integrity.

An AIB is a MIME body of type 'message/sipfrag' - for more information on constructing sipfrags, including examples, see [4]. This MIME body MUST have a Content-Disposition [3] disposition-type of 'aib', a new value defined in this document specifically for authenticated identity bodies. The Content-Disposition header SHOULD also contain a 'handling' parameter indicating that this MIME body is optional (i.e. if this mechanism is not supported by the user agent server, it can still attempt to process the request).

AIBs using the 'message/sipfrag' MIME type MUST contain the following headers when providing identity for an INVITE request: From, Date, Call-ID and Contact; they SHOULD also contain the To, and CSeq header. The security properties of these headers, and circumstances in which they should be used, are described in Section 10. AIBs MAY contain any other headers that help to uniquely identify the transaction or provide related reference integrity. An example of the AIB format for an INVITE is:

```
Content-Type: message/sipfrag
Content-Disposition: aib; handling=optional

From: Alice <sip:alice@example.com>
To: Bob <sip:bob@example.net>
Contact: <sip:alice@pc33.example.com>
Date: Thu, 21 Feb 2002 13:02:03 GMT
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
```

Unsigned AIBs MUST be treated by any recipients according to the rules set out in Section 7 for AIBs that do not validate. After the AIB has been signed, it SHOULD be added to any existing MIME bodies in the request (such as SDP), if necessary by transitioning the outermost MIME body to a 'multipart/mixed' format.

3. Example of a Request with AIB

The following shows a full SIP INVITE request with an AIB:

```
INVITE sip:bob@example.net SIP/2.0
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@example.net>
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.example.com>
Content-Type: multipart/mixed; boundary=unique-boundary-1

--unique-boundary-1

Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 example.com
s=Session SDP
c=IN IP4 pc33.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--unique-boundary-1
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42
```

```
Content-Length: 608

--boundary42
Content-Type: message/sipfrag
Content-Disposition: aib; handling=optional

From: Alice <sip:alice@example.com>
To: Bob <sip:bob@example.net>
Contact: <sip:alice@pc33.example.com>
Date: Thu, 21 Feb 2002 13:02:03 GMT
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
   handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--

--unique-boundary-1--
```

4. AIBs for Identifying Third-Parties

There are special-case uses of the INVITE method in which some SIP messages are exchanged with a third party before an INVITE is sent, and in which the identity of the third party needs to be carried in the subsequent INVITE. The details of addressing identity in such contexts are outside the scope of this document. At a high level, it is possible that identity information for a third party might be carried in a supplemental AIB. The presence of a supplemental AIB within a message would not preclude the appearance of a 'regular' AIB as specified in this document.

Example cases in which supplemental AIBs might appear include:

> The use of the REFER [5] method, for example, has a requirement for the recipient of an INVITE to ascertain the identity of the referrer who caused the INVITE to be sent.

Third-party call control (3PCC [6]) has an even more complicated identity problem. A central controller INVITEs one party, gathers identity information (and session context) from that party, and then uses this information to INVITE another party. Ideally, the controller would also have a way to share a cryptographic identity signature given by the first party INVITEd by the controller to the second party invited by the controller.

In both of these cases, the Call-ID and CSeq of the original request (3PCC INVITE or REFER) would not correspond with that of the request in by the subsequent INVITE, nor would the To and From. In both the REFER case and the 3PCC case, the Call-ID and CSeq cannot be used to guarantee reference integrity, and it is therefore much harder to correlate an AIB to a subsequent INVITE request.

Thus, in these cases some other headers might be used to provide reference integrity between the headers in a supplemental AIB with the headers of a 3PCC or REFER-generated INVITE, but this usage is outside of the scope of this document. In order for AIBs to be used in these third-party contexts, further specification work is required to determine which additional headers, if any, need to be included in an AIB in a specific third-party case, and how to differentiate the primary AIB in a message from a third-party AIB.

5. Identity in non-INVITE Requests

The requirements for populating an AIB in requests within a dialog generally parallel those of the INVITE: From, Call-ID, Date and Contact header fields are REQUIRED.

Some non-INVITE requests, however, may have different identity requirements. New SIP methods or extensions that leverage AIB security MUST identify any special identity requirements in the Security Considerations of their specification.

6. Identity in Responses

Many of the practices described in the preceding sections can be applied to responses as well as requests. Note that a new set of headers must be generated to populate the AIB in a response. The From header field of the AIB in the response to an INVITE MUST correspond to the address-of-record of the responder, NOT to the From header field received in the request. The To header field of the request MUST NOT be included. A new Date header field and Contact header field should be generated for the AIB in a response. The Call-ID and CSeq should, however, be copied from the request.

Generally, the To header field of the request will correspond to the

address-of-record of the responder. In some architectures where retargeting is used, however, this need not be the case. Some recipients of response AIBs may consider it a cause for security concern if the To header field of the request is not the same as the address-of-record in the From header field of the AIB in a response.

7. Receiving an AIB

When a user agent receives a request containing an AIB, it MUST verify the signature, including validating the certificate of the signer, and compare the identity of the signer (the subjectAltName) with, in the INVITE case, the domain portion of the URI in the From header field of the request (for non-INVITE requests, other headers MAY be subject to this comparison). The two should correspond exactly; if they do not, the user agent MUST report this condition to its user before proceeding. User agents MAY distinguish between plausibly minor variations (the difference between 'example.com' and 'sip.example.com') and major variations ('example.com' vs. 'example.org') when reporting these discrepancies in order to give the user some idea of how to handle this situation. Analysis and comparison of the Date, Call-ID and Contact header fields as described in Section 10 MUST also be performed. Any discrepancies or violations MUST be reported to the user.

When the originating user agent of a request receives a response containing an AIB, it SHOULD compare the identity in the From header field of the AIB of the response with the original value of the To header field in the request. If these represent different identities, the user agent SHOULD render the identity in the AIB of the response to its user. Note that a discrepancy in these identity fields is not necessary an indication of a security breach; normal retargeting may simply have directed the request to a different final destination. Implementors therefore may consider it unnecessary to alert the user of a security violation in this case.

8. Encryption of Identity

Many SIP entities that support the use of S/MIME for signatures also support S/MIME encryption, as described in RFC3261 Section 23.4.3.

While encryption of AIBs entails that only the holder of a specific key can decrypt the body, that single key could be distributed throughout a network of hosts that exist under common policies. The security of the AIB is therefore predicated on the secure distribution of the key. However, for some networks (in which there are federations of trusted hosts under a common policy), the widespread distribution of a decryption key could be appropriate. Some telephone networks, for example, might require this model.

When an AIB is encrypted, the AIB SHOULD be encrypted before it is signed. Implementations MUST still accept AIBs that have been signed and then encrypted.

9. Example of Encryption

The following is an example of an encrypted and signed AIB (without any of the preceding SIP headers). In a rendition of this body sent over the wire, the text wrapped in asterisks would be in ciphertext.

```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42
Content-Length: 568
Content-Disposition: aib; handling=optional

--boundary42

Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
  name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
  handling=required
Content-Length: 231

**********************************************************
* Content-Type: message/sipfrag                          *
* Content-Disposition: aib; handling=optional            *
*                                                        *
* From: sip:alice@example.com                            *
* Call-ID: a84b4c76e66710                                *
* Contact: sip:alice@device21.example.com                *
* Date: Thu, 21 Feb 2002 13:02:03 GMT                    *
**********************************************************

--boundary42

Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
   handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

10. Security Considerations

The purpose of an AIB is to provide an identity for the sender of a SIP message. This identity is held in the From header field of an AIB. While other headers are also included, they are provided solely to assist in detection of replays and cut-and-paste attacks leveraged to impersonate the caller. The contents of the From header field of a valid AIB are suitable for display as a "Caller ID" for the sender of the SIP message.

This document mandates the inclusion of the Contact, Date, Call-ID, and From header fields within an AIB, and recommends the inclusion of CSeq and To header fields, when 'message/sipfrag' is used to represent the identity of a request's sender. If these headers are omitted, some important security properties of AIB are lost. In general, the considerations related to the inclusion of various headers in an AIB are the same as those given in RFC3261 for including headers in tunneled 'message/sip' MIME bodies (see Section 23 in particular).

The From header field indicates the identity of the sender of the message; were this header to be exluced, the creator of the AIB essentially would not be asserting an identity at all. The Date and Contact headers provide reference integrity and replay protection, as described in RFC3261 Section 23.4.2. Implementations of this specification MUST follow the rules for acceptance of the Date header field in tunneled 'message/sip' requests described in RFC 3261 Section 23.4.2; this ensures that outdated AIBs will not be replayed (the suggested interval is that the Date header must indicate a time within 3600 seconds of the receipt of a message). Implementations MUST also record Call-IDs received in AIBs, and MUST remember those Call-IDs for at least the duration of a single Date interval (i.e. 3600 seconds). Accordingly, if an AIB is replayed witihn the Date interval, receivers will recognize that it is invalid because of a Call-ID duplication; if an AIB is replayed after the Date interval, receivers will recognize that it is invalid because the Date is stale. The Contact header field is included to tie the AIB to a particular device instance that generated the request. Were an active attacker to intercept a request containing an AIB, and cut-and-paste the AIB into their own request (reusing the From, Contact, Date and Call-ID fields that appear in the AIB), they would not be eligible to receive SIP requests from the called user agent, since those requests are routed to the URI identified in the Contact header field.

The To and CSeq header fields provide properties that are generally useful, but not for all possible applications of AIBs. If a new AIB is issued each time a new SIP transaction is initiated in a dialog,

then the CSeq header field provides a valuable property (replay protection for this particular transaction). If, however, one AIB is used for an entire dialog, subsequent transactions in the dialog would use the same AIB that appeared in the INVITE transaction. Using a single AIB for an entire dialog reduces the load on the generator of the AIB. The To header field usually designates the original URI that the caller intended to reach, and therefore it may vary from the Request-URI if retargeting occurs at some point in the network. Accordingly, including the To header field in the AIB helps to identify cut-and-paste attacks in which an AIB sent to a particular destination is reused to impersonate the sender to a different destination. However, the inclusion of the To header field probably would not make sense for many third-party AIB cases (as described in Section 4), nor is its inclusion necessary for responses.

11. IANA Considerations

This document defines a new MIME Content-Disposition disposition-type value of 'aib'. This value is reserved for MIME bodies that contain an authenticated identity, as described in section Section 2.

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, May 2002.

[2] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

[3] Troost, R., Dorner, S. and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

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

Informative References

[5] Sparks, R., "The SIP Refer Method", draft-ietf-sip-refer-07 (work in progress), November 2002.

[6] Rosenberg, J., Peterson, J., Schulzrinne, H. and G. Camarillo, "Best Current Practices for Third-Party Call Control in the Session Initiation Protocol", draft-ietf-sipping-3pcc-02 (work in progress), June 2002.

Author's Address

Jon Peterson
NeuStar, Inc.
1800 Sutter St
Suite 570
Concord, CA 94520
US

Phone: +1 925/363-8720
EMail: jon.peterson@neustar.biz
URI: http://www.neustar.biz/

Appendix A. Acknowledgements

The author would like to thank Robert Sparks, Jonathan Rosenberg, Mary Watson, and Eric Rescorla for their comments. Rohan Mahy also provided some valuable guidance.

Full Copyright Statement

Acknowledgement

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

SIP WG R. Mahy
Internet-Draft Cisco Systems, Inc.
Expires: January 15, 2005 July 17, 2004

Connection Reuse in the Session Initiation Protocol (SIP)
draft-ietf-sip-connect-reuse-02.txt

Status of this Memo

Copyright Notice

Abstract

When SIP entities use a connection oriented protocol to send a request, they typically originate their connections from an ephemeral port. The SIP protocol includes mechanisms which insure that responses to a request, and new requests sent in the original direction reuse an existing connection. However, new requests sent in the opposite direction are unlikely to reuse the existing connection. This frequently causes a pair of SIP entities to use one connection for requests sent in each direction, and can result in potential scaling and performance problems. This document proposes requirements and a mechanism which address this deficiency.

Table of Contents

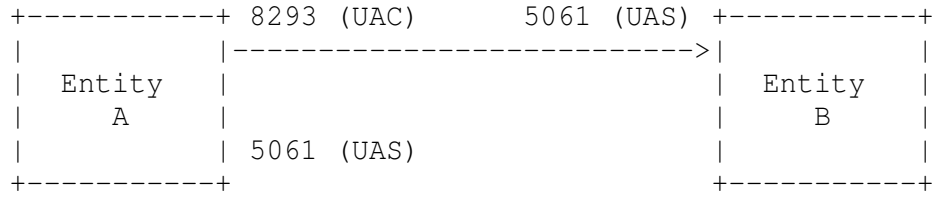
1. Conventions

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].

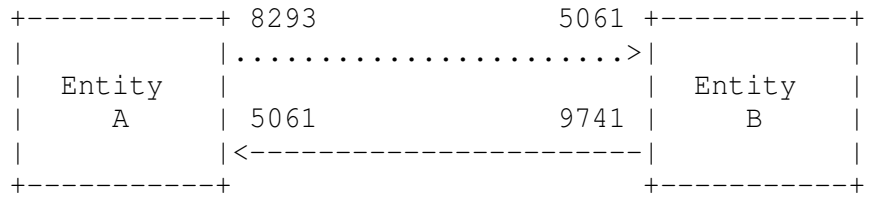
2. Introduction and Problem Statement

SIP [1] entities can communicate using either unreliable/ connectionless (ex: UDP) or reliable/connection-oriented (ex: TCP, SCTP [14]) transport protocols. When SIP entities use a connection-oriented protocol (such as TCP or SCTP) to send a request, they typically originate their connections from an ephemeral port.

In the following example, Entity A listens for SIP requests over TLS [4] on TCP port 5061 (the default port for SIP over TLS over TCP), but uses an ephemeral port (port 8293) for a new connection to Entity B. These entities could be SIP User Agents or SIP Proxy Servers.



The SIP protocol includes mechanisms which insure that responses to a request reuse the existing connection which is typically still available, and also includes provisions for reusing existing connections for other requests sent by the originator of the connection. However, new requests sent in the opposite direction (routed from the target of the original connection toward the originator of the original connection) are unlikely to reuse the existing connection. This frequently causes a pair of SIP entities to use one connection for requests sent in each direction, as shown below.



2.1 Efficiency Concerns

This extra pair of connections can result in potential scaling and

performance problems. For example, each new connection using TLS requires a TCP 3-way handshake, a handful of round-trips to establish TLS, and (typically) expensive asymetric authentication and key generation algorithms, and certificate verification. This effectively doubles the load on each entity. Setting up a second connection can also cause excessive delay (especially in networks with long round-trip times) for subsequent requests, even requests in the context of an existing dialog (for example a reINVITE or BYE after an initial INVITE, or a NOTIFY after a SUBSCRIBE [11] or a REFER [12]).

Consider the call flow shown below where Proxy A and Proxy B use the Record-Route mechanism to stay involved in a dialog. Proxy B will establish a new TLS connection just to send a BYE request.

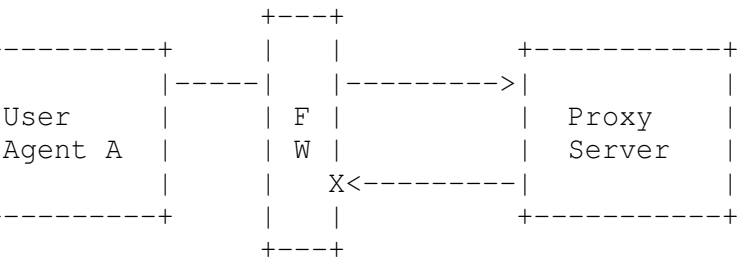
```
INVITE ->   create connection 1
<- 200      response over connection 1
ACK ->      reuse connection 1

<- BYE      create connection 2
-> 200      response over connection 2
```

ReINVITEs or UPDATE [8] requests are expected to be handled automatically and rapidly in order to avoid media and session state from being out of step. If a reINVITE requires a new TLS connection, the reINVITE could be delayed by several extra round-trip times. Depending on the round-trip time, this combined delay could be perceptible or even annoying to a human user. This is especially problematic for some common SIP call flows (for example, the recommended example flow in figure number 4 in RFC3725 [9] (SIP third-party call control) use many reINVITEs.

2.2 Directional Connectivity

Some SIP User Agents can initiate TCP or TLS connections, but for one reason or another cannot usefully accept incoming connections. When TLS connections are involved, many User Agents can accept incoming TLS connections, but cannot provide a certificate that is likely to be trusted by the TLS client. (The User Agent can only offer a self-signed certificate for example.) This cause their connectivity to fail

In other cases, a User Agent cannot accept incoming TCP connections at all because they are behind a Firewall or NAT.



Finally, some User Agents may be configured to refuse incoming TCP or TLS connections, since it is difficult or impossible for some class of User Agents to authorize such connections.

In all three of these cases, connection reuse is no longer simply an efficiency improvement. It is mandatory to make TCP or TLS work for a large class of SIP User Agents. For User Agents in this class, incoming requests need to come from a proxy server with which the User Agent has a persistent connection. Each User Agent in this class of UAs also needs to have a SIP URI with GRUU [7] properties that routes to a proxy server that knows how to forward requests over this persistent connection already opened by the User Agent. However, just using the GRUU mechanism by itself does not help a proxy server determine which incoming persistent connection is associated with a particular GRUU. An explicit connection reuse mechanism is still needed between these User Agents and their proxies.

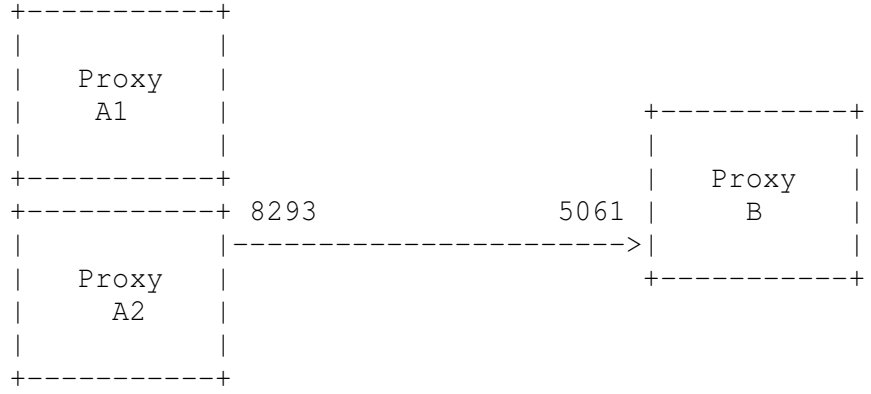
For example, consider a standard SIP presence [10] subscription. An instant messaging and presence client sends a SUBSCRIBE request for the presence of a peer. The presence notifications are returned as NOTIFY requests from the peer. If the subscribing User Agent wants to use TLS, the NOTIFY request would cause a second (new) connection back to the subscribing client. If the presence client doesn't have a valid TLS certificate anchored in a well-known certificate authority, the NOTIFY request will fail, and the presence client will never receive any presence data.

```
SUBSCRIBE ->          connection 1
<- 202                connection 1
<- NOTIFY             connection 2
200 ->                connection 2
```

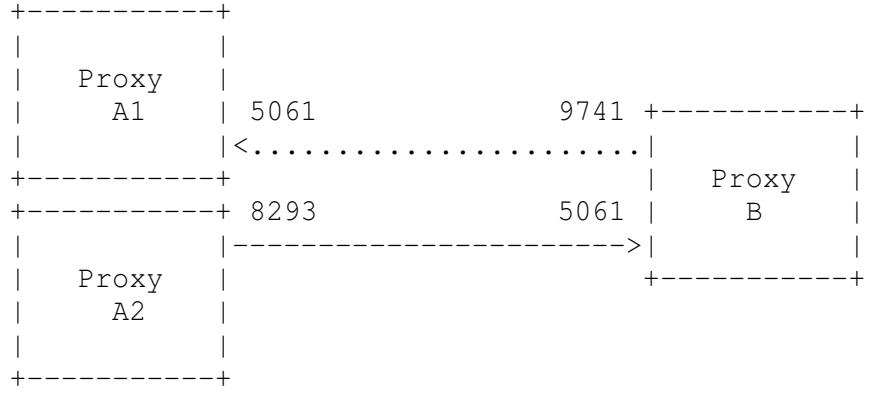
2.3 Clustering

When clusters or farms of cooperating SIP servers (for example proxy servers) are configured together, SIP entities have no way to prefer a server with an existing connection. For example, Proxy server B

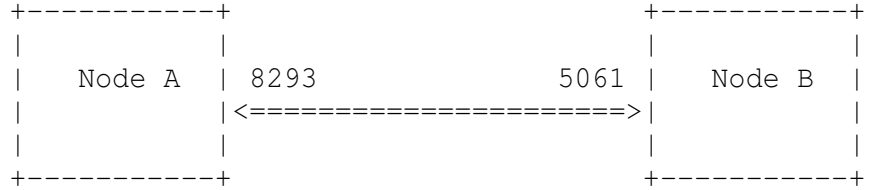
has no mechanism to choose an existing connection with Proxy cluster A.



As a result, Proxy B might open a new connection to another proxy server for requests sent in the opposite direction.



The rules for handling the Transport layer described in Section 18 of SIP [1] do not associate incoming connections with the listening port which corresponds to the same SIP entity. If the Tranport layer had some way to associate these connections, then request and responses originated from either node could reuse existing connections as shown below.



Likewise, when when an "outbound-only" user agent opens a connection to only one proxy server in the same cluster, there is no way for

this user agent to recieve incoming requests if the this particular proxy goes down. In many scenarios, the user agent would not notice the connection was unresponsive until it was time to refresh its SIP registration. Some "outbound-only" user agents would like the ability to open persistent connections to more than one proxy server in a cluster, so that at least two proxy servers in the cluster can forward incoming requests to the user agent.

2.4 Hop-by-hop reuse

All the examples presented above apply to a single hop as opposed to a per-dialog route-set or other complete path. This is natural since ordinary connections are managed on a hop-by-hop basis as well. In addition, it is useful to note that "outbound-only" user agents require connection reuse for any number of reuqests which may have nothing in common other than the last hop. Likewise, proxy server to proxy server connection reuse is likely to be much more efficient if multiple dialogs or multiple users can use the same connection.

3. Requirements

1. A connection sharing mechanism SHOULD allow SIP entities to reuse existing connections for requests and repsonses originated from either peer in the connection.
2. A connection sharing mechanism SHOULD allow SIP entities to reuse existing connections with closely coupled nodes which act as a single SIP entity (for example a cluster of nodes acting as a proxy server).
3. A connection sharing mechanism MUST NOT require UACs (clients) to send all traffic from well-know SIP ports.
4. A connection sharing mechanism MUST NOT require configuring ephemeral port numbers in DNS.
5. A connection sharing mechanism MUST prevent unauthorized hijacking of other connections.
6. Connection sharing SHOULD persist across SIP transactions and dialogs.
7. There is no requirement to share a complete path. Hop-by-hop connection sharing is more appropriate.

4. Behavior

The proposed mechanism uses a new Via header field parameter. The "alias" parameter is included in a Via header field value to indicate that the originator of the request wants to create a transport layer alias. The originator places their alias in the Via header field value (in the "sent-by" production). This "alias" address becomes mapped to the a actual IP address and port number observed as the source address of the current connection.

Assuming the Via header field value shown below from the most recent request arrived over a connection from 10.54.32.1 port 8241:

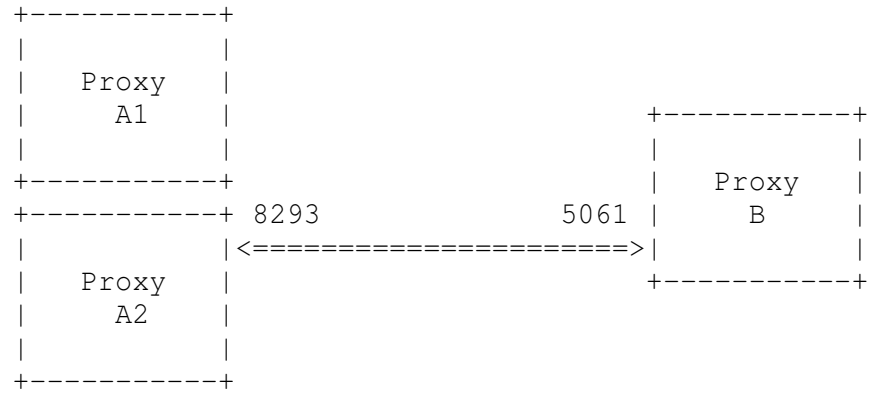
```
Via: SIP/2.0/TLS 10.54.32.1:5061;branch=z9hG4bKa7c8dze ;alias
```

The transport layer creates an alias, such that any requests going to the "advertised address" (10.54.32.1 port 5061) are instead sent over the existing connection (to the "target" of the alias) which is coming from port 8241. This sharing continues as long as the target connection stays up.

The SIP community recommends that servers keep connections up unless they need to reclaim resources, and that clients keep connections up as long as they are needed. Connection reuse works best when the client and the server maintain their connections for long periods of time. SIP entities therefore SHOULD NOT drop connections on completion of a transaction or termination of a dialog.

To implement connection aliases for explicit IP addresses and port numbers, a SIP node could (for example) search an additional data structure (the alias table) prior to opening a new connection, or could modify the data structure in which it keeps active connection state so that aliases, active connnections, and blacklisted nodes are all discovered when looking for an active connection.

Likewise when clusters or farms of cooperating SIP servers (for example proxy servers) are configured together, this mechanism allows a SIP entity to select a server with an existing connection. With this mechanism, Proxy B sends requests for Proxy cluster A to node A2 with whom it already shares an existing connection.



For example, on receipt of a message with the topmost Via header shown below, the transport layer creates an alias such that requests going to the advertised address (proxy-farm-a.example.com) are sent over the target connection (from 10.54.32.1:8241).

```
Via: SIP/2.0/TLS proxy-farm-a.example.com;branch=z9hG4bK7c8ze;alias
```

As a result, this has an important interaction with the DNS resolution mechanisms for SIP described in RFC3263 [6]. When new requests arrive for proxy-farm-a.example.com, proxy B still needs to perform a DNS NAPTR lookup to select the transport. Once the transport is selected, an SRV lookup would ordinarily occur to find the appropriate port number. In this case, the transport layer uses a connection reuse alias instead of performing the SRV query.

Below is a partial DNS zone file for atlanta.com.

```
; NAPTR queries for the current domain (example.com)
;
; order pref flags service regexp replacement
proxy-farm-a IN NAPTR 50 50 "s" "SIPS+D2T" "" _sips._tcp.

; SRV records for the proxy use 5060/5061
;
;; Priority Weight Port Target
_sips._tcp.proxy-farm-a  IN SRV 0 1 5061 host-a1
_sips._tcp.proxy-farm-a  IN SRV 0 1 5061 host-a2

host-a1  IN A 10.54.32.1
host-a2  IN A 10.54.32.2
```

The existence of an alias parameter in a Via header in a request is treated as a request to the transport layer to create an alias (named by the sent-by parameter) that points to the alias target (the current connection)

This mechanism is fully backwards compatible with existing implementations. If the proposed Via parameter is not understood by the recipient, it will be ignored and the two implementations will revert to current behavior (two connections).

4.1 Authorizing an alias request

Authorizing connection aliases is essential to prevent connection hijacking. For example a program run by a malicious user of a multiuser system could attempt to hijack SIP requests destined for the well-known SIP port from a large relay proxy.

To correctly authorize an alias, the SIP node authorizing the request needs to recognize both the active connection and the alias as the same resource. The only way to accomplish this is if both the active connection and the alias can be authenticated using the same credentials. This could be accomplished using one of two mechanisms.

4.1.1 Authorizing using TLS mutual authentication

The first (and preferred) authorization mechanism is using TLS mutual authentication, such that the subjectAltName of the originator certificate corresponds to both the current connection and the target address of the alias. The Via sent-by address needs to be within the scope protected by the certificate presented by the originator during TLS mutual authentication and the received IP address needs be a valid IP address for the sent-by host or hosts. In other words, the sent-by address MUST be resolvable from the subjectAltName of the originator certificate, and the received IP address MUST be resolvable from the sent-by address. This is in addition to other requirements for TLS authentication and authorization discussed in SIP [1] and Locating SIP Servers [6].

Following this logic step-by-step:

1. Verify that the certificate presented is not expired and is rooted in a trusted certificate chain.
2. Verify that the subjectAltName in the certificate covers the "advertised address" (the address in the Via sent-by production). If the advertised address and the subjectAltName match exactly then the certificate covers the address.
3. Use DNS to resolve if the advertised name is resolvable from the subjectAltName (start by resolving the subjectAltName as if it where a target URI according to the rules in RFC3263. That is, if no port number is present perform an SRV lookup, then finally resolve relevant address records). If any of the resolved addresses (port numbers can be ignored in this case) matches the advertised address, then the certificate covers the address.
4. Finally, Verify that the advertised address can resolve to the IP address over which the connection was received.

For example, take the example in the previous section of proxy B receiving an alias request from host-a2.example.com. Proxy B verifies that the presented certificate is valid and trusted. Proxy B checks that proxy-farm-a.example.com is both the advertised name and the subjectAltName in the certificate. Finally, proxy B verifies that this connection is coming from 10.54.32.2, which is one of the addresses in DNS for host-a2.example.com, which in turn is mentioned in an SRV record for _sips._tcp.proxy-farm-a.example.com.

4.1.2 Authorization via Registration

The second mechanism is to accept an alias if the target address of the alias is equivalent (using SIP comparison rules) to a valid Contact already registered by the same user. This user could be authenticated through any SIP or TLS mechanism (ex: user certificate, or Kerberos [13]), but would typically use Digest authentication [5].

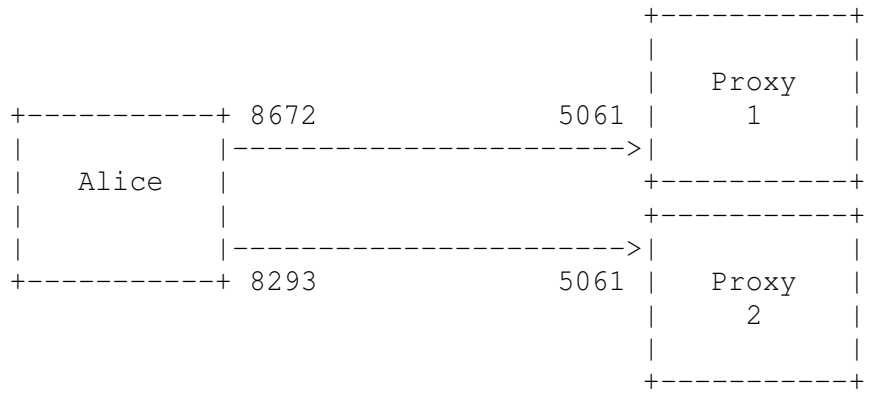
For example, if Alice registers a Contact of 198.168.67.89:5061, she could inform Proxy 1 of the existence of a connection to her from Proxy 2. This would allow her to preemptively originate TLS connections, as her user agent may not have access to a site certificate with which to authenticate incoming TLS connections.

The Proxy takes the following steps to authorize these requests:

1. The Proxy authenticates or authorizes the sender for an otherwise ordinary SIP request.
2. The Proxy looks for any Contacts in the location/registration service which have a hostport and transport that matches exactly the advertised address.
3. The Proxy checks if the user who sent the request would be authorized to change the Contact found when looking up the Contact URI in the location/registration service.

For example, Alice advertises the address "198.168.67.89:5061" in the Via header field of a request sent over a connection from "198.168.67.89:8293" to Proxy 2. The Proxy otherwise authenticates Alice's request (for example an INVITE request). The Proxy looks up 198.168.67.89:5061 and finds the following Contact: "Alice" <sips:reg2@198.168.67.89:5061>. Alice is authorized to modify Alice's contact, so Alice is authorized to alias an advertised address "reserved" by one of her Contacts. Alice then sends another request (this time an OPTIONS request for example) to Proxy 1 from "198.168.67.89.8672" with the same Via header. Proxy 1 similarly authorizes Alice's request and stores the alias. Now if either proxy receives a request for 198.168.67.89:5061, it will forward the request over the appropriate existing connection with Alice.

```
Via: SIP/2.0/TLS 198.168.67.89:5061;branch=z9hG4bK7c8dze ;alias
```



4.2 Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [3]. This document proposes to extend via-params to include a new via-alias defined below.

```
via-params = via-ttl / via-maddr / via-received / via-branch /
             via-alias / via-extension

via-alias  = "alias"
```

5. Security Considerations

This document presents requirements and a mechanism for reusing existing connections easily. Unauthorized connection reuse would present many opportunities for rampant abuse and hijacking, but these attacks can be prevented if the guidelines in Section 4.1 are followed.

The mechanism in this document can significantly improve the security of SIP networks, in that it makes SIP over TLS (and the sips: scheme) practical for devices which do not have a certificate rooted in a well-known certificate authority.

SIP Proxy Servers which implement this mechanism MUST implement TLS mutual authentication. Digest authentication is already mandatory to implement for all SIP implementations.

6. IANA Considerations

This document adds a parameter to the SIP header field parameters registry:

Header field in which parameter can appear: Via
Name of the parameter: alias
Reference: This document

7. Acknowledgments

Thanks to Jon Peterson for helpful answers about certificate behavior with SIP, Jonathan Rosenberg for his initial support of this concept, and Cullen Jennings for providing a sounding board for this idea.

8. References

8.1 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] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[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] 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.

[6] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[7] 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 Informational References

[8] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[9] Rosenberg, J., Peterson, J., Schulzrinne, H. and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

[10] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", draft-ietf-simple-presence-10 (work in progress), January 2003.

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

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

[13] Kohl, J. and B. Neuman, "The Kerberos Network Authentication

Service (V5)", RFC 1510, September 1993.

[14] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

Author's Address

Rohan Mahy
Cisco Systems, Inc.
5617 Scotts Valley Dr
Scotts Valley, CA 95066
USA

EMail: rohan@cisco.com

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

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

D. Willis, Ed.
dynamicsoft Inc.

A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages
draft-ietf-sip-content-indirect-mech-04

Status of this Memo

Copyright Notice

Abstract

This document proposes an extension to the URL MIME External- Body Access-Type to satisfy the content indirection requirements for SIP. These extensions are aimed at allowing any MIME part in a SIP message to be referred to indirectly via a URI.

Table of Contents

1. Terminology

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 [5]

2. Introduction

The purpose of the Session Initiation Protocol [9] (SIP) is to create, modify, or terminate sessions with one or more participants. SIP messages, like HTTP, are sytnactically composed of a start line, one or more headers, and an optional body. Unlike HTTP, SIP is not designed as a general purpose transport of data.

There are numerous reasons why it might be desirable to indirectly specify the content of the SIP message body. For bandwidth limited applications such as cellular wireless, indirection provides a means to annotate the (indirect) content with meta-data which may be used by the recipient to determine whether or not to retrieve the content over the resource limited link.

It is also possible that the content size to be transferred might potentially overwhelm intermediate signaling proxies, thereby unnecessarily increasing network latency. For time-sensitive SIP applications, this may be unacceptable. Indirect content can remedy this by moving the transfer of this content out of the SIP signaling network and into a potentially separate data transfer channel.

There may also be scenarios where the session related data (body) that needs to be conveyed does not directly reside on the endpoint or User Agent. In such scenarios, it is desirable to have a mechanism whereby the SIP message can contain an indirect reference to the desired content. The receiving party would then use this indirect reference to retrieve the content via a non-SIP transfer channel such as HTTP, FTP, or LDAP.

The purpose of content indirection is purely to provide an alternative transport mechanism for SIP MIME body parts. With the exception of the transport mechanism, indirected body parts are equivalent, and should have the same treatment, as in-line body parts.

Previous attempts at solving the content indirection problem made use of the text/uri-list [6] MIME type. While attractive for its simplicity (a list of URIs delimted by end-of-line markers), it fails to satisfy a number of the requirements for a more general purpose content indirection mechanism in SIP. Most notably lacking is the ability to specify various attributes on a per-URI basis. These

attributes might include version information, the MIME type of the referenced content, etc.

In searching for a replacement for the text/uri-list MIME type, RFC2017 defines a strong candidate. RFC2017 [1] defines an extension to the message/external-body MIME type originally defined in RFC2046 [3]. The extension that RFC2017 makes is to allow a generic URI to specify the location of the content rather than protocol specific parameters for FTP, etc. as originally defined in RFC2046. While providing most of the functionality needed for a SIP content indirection mechanism, RFC2017 by itself is not a complete solution. This document will specify the usage of RFC2017 necessary to fulfill the requirments outlined for content indirection.

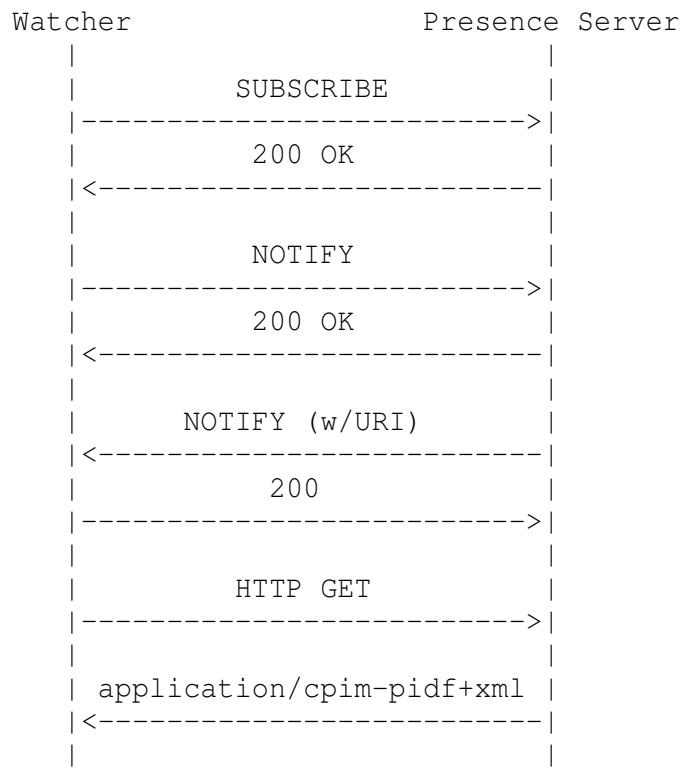
The requirements can be classified as applying either to the URI which indirectly references the desired content or to the content itself. Where possible, existing MIME parameters and entity headers are used to satisfy those requirements. MIME (Content-Type) parameters are the preferred manner of describing the URI while entity headers are the preferred manner of describing the (indirect) content. See RFC 2045 [2] for a description of most of these entity headers and MIME parameters.

3. Example Use Cases

There are several example users of such a content indirection mechanism. These are examples only and are not intended to limit the scope or applicability of the mechanism.

3.1 Presence Notification

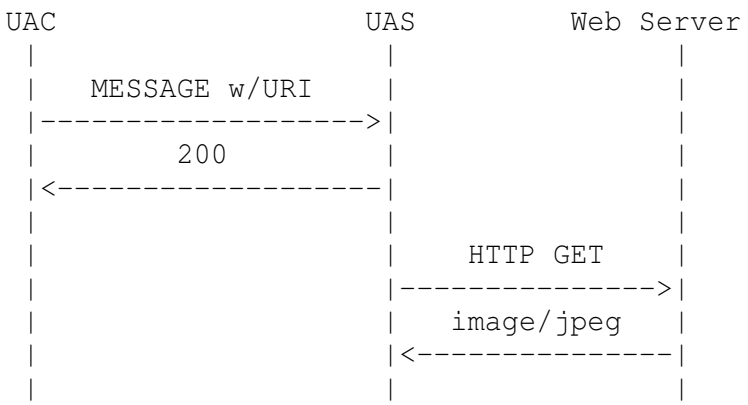
The information carried in a presence document could potentially exceed the recommended size for a SIP (NOTIFY) request, particularly if the document carries aggregated information from multiple endpoints. In such a situation, it would be desirable to send the NOTIFY request with an indirect pointer to the presence document which could then be retrieved by, for example, HTTP.



In this example, the presence server returns an HTTP URI pointing to a presence document on the presence server which the watcher can then fetch using an HTTP GET.

3.2 Document Sharing

During an instant messaging conversation, a useful service is document sharing wherein one party sends an IM (MESSAGE request) with an indirect pointer to a document which is meant to be rendered by the remote party. Carrying such a document directly in the MESSAGE request is not appropriate for most documents. Furthermore, the document to be shared may reside on a completely independent server from the originating party.



In this example, a user wishes to exchange a JPEG image that she has stored on her web server with another user she has a IM conversation with. The JPEG is intended to be rendered inline in the IM conversation. The recepient of the MESSAGE request launches a HTTP GET request to the web server to retrieve the JPEG image.

4. Requirements

- o It MUST be possible to specify the location of content via a URI. Such URIS MUST be conformnt with RFC2396 [7] or its successors, such as [10].
- o It MUST be possible to specify the length of the indirect content.
- o It MUST be possible to specify the type of the indirect content.
- o It MUST be possible to specify the disposition of each URI independently.
- o It MUST be possible to label each URI to identify if and when the content referred to by that URI has changed. Applications of this mechanism may send the same URI more than once. The intention of this requirement is to allow the receiving party to determine if the content referenced by the URI has changed without having to actually retrieve that content. Example ways the URI could be labelled include a sequence number, timestamp, version number, etc. When used with HTTP, an entity-tag (ETAG) mechanism as defined in RFC2068 [4]" may be appropriate. Note that we are not labeling the URI itself, but the content to which the URI refers, and that the label is therefore effectively "metadata" of the content itself.
- o It MUST be possible to specify the timespan for which a given URI is valid. This may or may not be the same as the lifetime for the content itself.
- o It MUST be possible for the UAC and the UAS to indicate support of this content indirection mechanism. A fallback mechanism SHOULD be specified in the event that one of the parties is unable to support content indirection.

- It MUST be possible for the UAC and UAS to negotiate the type of the indirect content when using the content indirection mechanism.
- It MUST be possible for the UAC and UAS to negotiate support for URI scheme(s) to be used in the content indirection mechanism. This is in addition to the ability to negotiate the content type.
- It SHOULD be possible to ensure the integrity and privacy of the URI when it is received by the remote party.
- It MUST be possible to process the content indirection without human intervention.
- It MUST allow for indirect transference of content in any SIP message which would otherwise carry that content as a body.

5. Application of RFC2017 to the Content Indirection Problem

The following text describes the application of RFC2017 to the requirements for content indirection.

5.1 Specifying support for content indirection

A UAC/UAS may indicate support for content indirection through an Accept header containing the message/external-body MIME type. The UAC/UAS must supply additional values in the Accept header to indicate the content types that it is willing to accept either directly or through content indirection. User-Agents supporting content indirection MUST support content indirection of the application/sdp MIME type.

For example:

```
Accept: message/external-body, image/*, application/sdp
```

5.2 Mandatory support for HTTP URI

Applications which use this content indirection mechanism MUST support at least the HTTP URI scheme. Additional URI schemes MAY be used, but a UAC/UAS MUST support receiving a HTTP URI for indirect content if it advertises support for content indirection.

The intention is to establish a baseline of support to further strengthen interoperability. Implementors may design for the most common case (HTTP) without having to worry about negotiation of support for this particular URI scheme.

5.3 Rejecting content indirection

If a UAS receives a SIP request which contains a content indirection

payload, and the UAS cannot or does not wish to support such a content type, it MUST reject the request with a 415 Unsupported Media Type response as defined in section 21.4.13 of SIP [9]. In particular, the UAC should note the absence of the message/external-body MIME type in the Accept header of this response to indicate that the UAS does not support content indirection.

5.4 Specifying the location of the content via a URI

The URI for the indirect content is specified in a "URI" parameter of the message/external-body MIME type. An access-type parameter indicates that the external content is referenced by a URI.

For example:

```
Content-Type: message/external-body;
              access-type="URL";
              URL="http://www.example.com/the-indirect-content"
```

5.5 Specifying versioning information for the URI

In order to determine whether or not the content indirectly referenced by the URI has changed, a Content-ID entity header is used. The syntax of this header is defined in RFC2045 [2]. Changes in the underlying content referred to by a URI MUST result in a change in the Content-ID associated with that URI. Multiple SIP messages carrying URI that refer to the same content SHOULD reuse the same Content-ID to allow the receiver to cache this content and avoid unnecessary retrievals. The Content-ID is intended to be globally unique and SHOULD be temporally unique across SIP dialogs.

For example:

```
Content-ID: <4232423424@www.example.com>
```

5.6 Specifying the lifetime of the URI

The URI supplied by in Content-Type header is not required to be accessible or valid for an indefinite period of time. Rather, the supplier of the URI MUST specify the time period for which this URI is valid and accessible. This is done through an "EXPIRATION" parameter of the Content-Type. The format of this expiration parameter is a RFC1123 date-time value. This is further restricted in this application to use only GMT time, consistent with the Date:

header in SIP. This is a mandatory parameter. Note that the date-time value can range from minutes to days or even years.

For example:

```
Content-Type: message/external-body;
              expiration="Mon, 24 June 2002 09:00:00 GMT"
```

5.7 Specifying the type of the indirect content

To support existing SIP mechanisms for the negotiation of content types, a Content-Type entity header SHOULD be present in the entity (payload) itself. If the protocol (scheme) of the URI supports its own content negotiation mechanisms (e.g. HTTP), this header may be omitted. The sender MUST however be prepared for the receiving party to reject content indirection if the receiver is unable to negotiate an appropriate MIME type using the underlying protocol for the URI scheme.

For example:

```
Content-Type: message/external-body; access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
              URL="http://www.example.com/the-indirect-content"
<CRLF>
Content-Type: application/sdp
<CRLF>
```

5.8 Specifying the size of the indirect content

When known in advance, the size of the indirect content should be supplied via a size parameter on the Content-Type header. This is an extension of RFC2017 but in line with other access types defined for the message/external-body MIME type in RFC2046. The content size is useful for the receiving party to make a determination about whether or not to retrieve the content. As with directly supplied content, a UAS may return a 513 error response in the event the content size is too large. This is an optional parameter.

For example:

```
Content-Type: message/external-body; access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
              URL="http://www.example.com/the-indirect-content";
              size=4123
```

5.9 Specifying the purpose of the indirect content

A Content-Disposition entity header SHOULD be present for all indirect content. In the absence of an an explicit Content-Disposition header, a content disposition of "session" should be assumed.

For example:

```
Content-Type: message/external-body; access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
              URL="http://www.example.com/the-indirect-content"
<CRLF>
Content-Type: image/jpeg
Content-Disposition: render
```

5.10 Specifying multiple URIs for content indirection

If there is a need to send multiple URIs for the purpose of content indirection, an appropriate multipart MIME type [3] should be used. Each URI should be contained in a single entity. Indirect content may be mixed with directly supplied content. This is particularly useful with the multipart/alternative MIME type.

For example:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=boundary42

--boundary42
Content-Type: text/plain; charset=us-ascii

The company announcement for June, 2002 follows:
--boundary42
Content-Type: message/external-body;
              access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
URL="http://www.example.com/announcements/07242002";
size=4123

Content-Type: text/html
Content-Disposition: render

--boundary42--
```

5.11 Specifying a hash value for the indirect content

If the specific content being referenced by the indirection is known to the sender, and the sender wishes the recipient to be able to validate that this content has not been altered from that intended by the sender, the sender includes a SHA-1 [8] hash of the content. If included, the hash is encoded by extending the MIME syntax [3] to include a "hash" parameter for the content type "message/external-body", the value of which is a base-64 enoding of the hash.

For example:

```
Content-Type: message/external-body;
              access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
              URL="http://www.example.com/the-indirect-content.au";
              size=52723
              hash=10AB568E91245681AC1B
<CRLF>
```

5.12 Supplying additional comments about the indirect content

Optional, freeform text may be supplied to comment on the indirect content. This should be supplied in a Content-Description entity header. This text may be displayed to the end user but MUST NOT used by other elements to determine disposition of the body, as such as usage would result in unreviewed extension to the COntent-type and Content-disposition header field functions.

For example:

```
Content-Type: message/external-body;
              access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
              URL="http://www.example.com/the-indirect-content";
              size=52723
<CRLF>
Content-Description: Multicast gaming session
```

5.13 Relationship to Call-Info, Error-Info, and Alert-Info Headers

SIP [9] defines three headers which are used to supply additional information with regard to a session, a particular error response, or alerting. All three of these headers allow the UAC or UAS to indicate additional information through a URI. They may be considered a form of content indirection. The content indirection mechanism defined in this document is not intended as a replacement for these headers. Rather, the headers defined in SIP MUST be used in preference to this mechanism where applicable because of the well defined semantics of those headers.

6. Examples

6.1 Single Content Indirection

```
INVITE sip:boromir@example.com SIP/2.0
From: <sip:gandalf@nwt.com>;tag=347242
To: <sip:boromir@example.com>
Call-ID: 3573853342923422@nwt.com
CSeq: 2131 INVITE
Accept: message/external-body application/sdp
Content-Type: message/external-body;
              ACCESS-TYPE=URL;
              URL="http://www.nwt.com/party/06/2002/announcement";
EXPIRATION="Sat, 20 Jun 2002 12:00:00 GMT"
size=231
Content-Length: ...

Content-Type: application/sdp
Content-Disposition: session
Content-ID: <4e5562cd1214427d@nwt.com>
```

6.2 Multipart MIME with Content Indirection

```
MESSAGE sip:boromir@example.com SIP/2.0
From: <sip:gandalf@nwt.com>;tag=34589882
To: <sip:boromir@example.com>
Call-ID: 92428924422111l7@nwt.com
CSeq: 388 MESSAGE
Accept: message/external-body, text/html, text/plain,
        image/*, text/x-emoticon
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=zz993453

--zz993453
Content-Type: message/external-body;
              access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
URL="http://www.nwt.com/company_picnic/image1.png"
size=234422

Content-Type: image/png
Content-ID: <9535035333@nwt.com>
Content-Disposition: render
Content-Description: Kevin getting dunked in the wading pool

--zz993453
Content-Type: message/external-body;
              access-type="URL";
              expiration="Mon, 24 June 2002 09:00:00 GMT";
URL="http://www.nwt.com/company_picnic/image2.png"
size=233811

Content-Type: image/png
Content-ID: <1134299224244@nwt.com>
Content-Disposition: render
Content-Description: Peter on his tricycle

--zz993453--
```

7. Security Considerations

Any content indirection mechanism introduces additional security concerns. By its nature, content indirection requires an extra processing step and information transfer. There are a number of potential abuses of a content indirection mechanism:

- Content indirection allows the initiator to choose an alternative protocol with weaker security or known vulnerabilities for the content transfer. For example, asking the recipient to issue an

HTTP request which results in a Basic authentication challenge.

- o Content indirection allows the initiator to ask the recipient to consume additional resources in the information transfer and content processing, potentially creating an avenue for denial of service attacks. For example, an active FTP URL consuming 2 connections for every indirect content message.
- o Content indirection could be used as a form of port scanning attack where the indirect content URL is actually a bogus URL pointing to an internal resource of the recipient. The response to the content indirection request could reveal information about open (and vulnerable) ports on these internal resources.
- o A content indirection URL can disclose sensitive information about the initiator such as an internal user name (as part of an HTTP URL) or possibly geolocation information.

Fortunately, all of these potential threats can be mitigated through careful screening of both the indirect content URIs that are received as well as those that are sent. Integrity and privacy protection of the indirect content URI can prevent additional attacks as well.

For confidentiality, integrity, and authentication, this content indirection mechanism relies on the security mechanisms outlined in RFC3261. In particular, the usage of S/MIME as defined in section 23 of RFC3261 provides the necessary mechanism to ensure integrity protection and privacy of the indirect content URI and associated parameters.

Securing the transfer of the indirect content is the responsibility of the underlying protocol used for this transfer. If HTTP is used, applications implementing this content indirection method MUST support the HTTPS URI scheme for secure transfer of content and must support the upgrading of connections to TLS using starttls. Note that a failure to complete HTTPS or starttls (for example, due to cert or encryption mismatch) after having accepted the indirect content in the SIP request is not the same as rejecting the SIP request, and may require additional user-user communication for correction.

Access control to the content referenced by the URI is not defined by this specification. Access control mechanisms may be defined by the protocol for the scheme of the indirect content URI.

If the UAC knows the content in advance, the UAC SHOULD include a hash parameter in the content indirection. The hash parameter is a base64-encoded SHA-1 hash of the indirected content. [8] If a hash value is included, the recipient MUST check the indirect content against that hash and indicate any mismatch to the user.

In addition, if the hash parameter is included, and the target URI involves setting up a security context using certificates, the UAS MUST ignore the results of the certificate validation procedure, and instead verify that the hash of the (canonicalized) content received matches the hash presented in the content-indirection hash parameter.

If the hash parameter is NOT included, the sender SHOULD use only schemes which offer message integrity (such as https:). When the hash parameter is not included and security using certificates is used, the UAS MUST verify any server certificates using the UAS's list of trusted top-level certificate authorities.

If hashing of indirected content is not used, the possibility exists that the content returned to the recipient by exercise of the indirection has been altered from that intended by the sender.

8. IANA Considerations

This document raises no new IANA considerations.

9. Contributions

It should be noted that the vast majority of this document, including editorship through the first IESG review, was provided by Sean Olson, seanol@microsoft.com

10. References

10.1 Normative References

[1] Freed, N. and K. Moore, "Definition of the URL MIME External-Body Access-Type", RFC 2017, October 1996.

[2] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[3] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[4] Fielding, R., Gettys, J., Mogul, J., Nielsen, H. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

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

[6] Daniel, R., "A Trivial Convention for using HTTP in URN Resolution", RFC 2169, June 1997.

[7] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[8] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[9] 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.

10.2 References References

[10] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", draft-fielding-uri-rfc2396bis-05 (work in progress), April 2004.

Author's Address

Dean Willis (editor)
dynamicsoft Inc.

EMail: dean.willis@softarmor.com
URI: http://www.softarmor.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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

SIP
Internet-Draft
Expires: December 31, 2004

J. Rosenberg
dynamicsoft

July 2, 2004

Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)
draft-ietf-sip-gruu-02

Status of this Memo

Copyright Notice

Abstract

Several applications of the Session Initiation Protocol (SIP) require a user agent (UA) to construct and distribute a URI which can be used by anyone on the Internet to route a call to that specific UA instance. A URI which routes to a specific UA instance is called a Globally Routable UA URI (GRUU). This document describes an extension to SIP for obtaining a GRUU from a server, and for communicating a GRUU to a peer within a dialog.

Table of Contents

1. Introduction

Several applications of the Session Initiation Protocol (SIP) [1] require a user agent (UA) to construct and distribute a URI which can be used by anyone on the Internet to route a call to that specific UA instance. An example of such an application is call transfer [18], based on the REFER method [5]. Another application is the usage of endpoint-hosted conferences within the conferencing framework [14]. We call these URIs Globally Routable UA URIs (GRUU). This specification provides a mechanism for obtaining and using GRUUs.

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 [3] and indicate requirement levels for compliant implementations.

3. Defining a GRUU

A GRUU is a SIP URI which has two characteristics:

Global: It can be used by any UAC connected to the Internet. In that regard, it is like an address-of-record (AOR) for a user. The address-of-record for a user, sip:joe@example.com, is meant to be used by anyone to reach that user. The same is true for a GRUU.

Routes to a Single Instance: It routes to a specific UA instance, and never forks. In that regard, it is unlike an address-of-record. When a request is sent to a normal AOR which represents a user, routing logic is applied in proxies to deliver the request to one or more UAs. That logic can result in a different routing decision based on the time-of-day, or the identity of the caller. However, when a request is made to a GRUU, the routing logic is dictated by the properties of a GRUU. The request has to be delivered to a very specific UA instance. That UA instance has to be the same UA instance for all requests sent to that GRUU. This does not mean that a GRUU represents a fundamentally different type of URI; it only means that the logic a proxy applies to a GRUU is going to generally be simpler than that it applies to a normal AOR.

4. Use Cases

We have encountered several use cases for a GRUU.

4.1 REFER

Consider a blind transfer application [18]. User A is talking to

user B. User A wants to transfer the call to user C. So, user A sends a REFER to user C. That REFER looks like, in part:

```
REFER sip:C@example.com SIP/2.0
From: sip:A@example.com;tag=99asd
To: sip:C@example.com
Refer-To: (URI that identifiers B's UA)
```

The Refer-To header field needs to contain a URI that can be used by user C to place a call to user B. However, this call needs to route to the specific UA instance which user B is using to talk to user A. If it didn't, the transfer service would not execute properly. This URI is provided to user A by user B. Because user B doesn't know who user A will transfer the call to, the URI has to be usable by anyone. Therefore, it is a GRUU.

4.2 Conferencing

A similar need arises in conferencing [14]. In that framework, a conference is described by a URI which identifies the focus of the conference. The focus is a SIP UA that acts as the signaling hub for the conference. Each conference participant has a dialog with the focus. One case described in the framework is where a user A has made a call to user B. User A puts user B on hold, and calls user C. Now, user A has two separate dialogs for two separate calls - one to user B, and one to user C. User A would like to conference them. To do this, user A's user agent morphs itself into a focus. It sends a re-INVITE or UPDATE [2] on both dialogs, and provides user B and user C with an updated Contact URI that now holds the conference URI. The Contact URI also has a callee capabilities [9] parameter which indicates that this URI is a conference URI. User A proceeds to mix the media streams received from user B and user C. This is called an ad-hoc conference.

At this point, normal conferencing features can be applied. That means that user B can send another user, user D, the conference URI, perhaps in an email. User D can send an INVITE to that URI, and join the conference. For this to work, the conference URI used by user A in its re-INVITE or UPDATE has to be usable by anyone, and it has to route to the specific UA instance of user A that is acting as the focus. If it didn't, basic conferencing features would fail. Therefore, this URI is a GRUU.

4.3 Presence

In a SIP-based presence [19] system, the Presence Agent (PA) generates notifications about the state of a user. This state is represented with the Presence Information Document Format (PIDF)

[17]. In a PIDF document, a user is represented by a series of tuples, each of which describes the services that the user has. Each tuple also has a contact URI, which is a SIP URI representing that device. A watcher can make a call to that URI, with the expectation that the call is routed to the service whose presence is represented in the tuple.

In some cases, the service represented by a tuple may exist on only a single user agent associated with a user. In such a case, the URI in the presence document has to route to that specific UA instance. Furthermore, since the presence document could be used by anyone who subscribes to the user, the URI has to be usable by anyone. As a result, it is a GRUU.

It is interesting to note that the GRUU may need to be constructed by a presence agent, depending on how the presence document is computed by the server.

5. Overview of Operation

This section is tutorial in nature, and does not specify any normative behavior.

This extension allows a UA to obtain a GRUU, and to use a GRUU. These two mechanisms are separate, in that a UA can obtain a GRUU in any way it likes, and use the mechanisms in this specification to use them. Similarly, a UA can obtain a GRUU but never use it. This specification defines two mechanisms for obtaining a GRUU - through registrations, and through administrative operation. Only the former requires protocol operations.

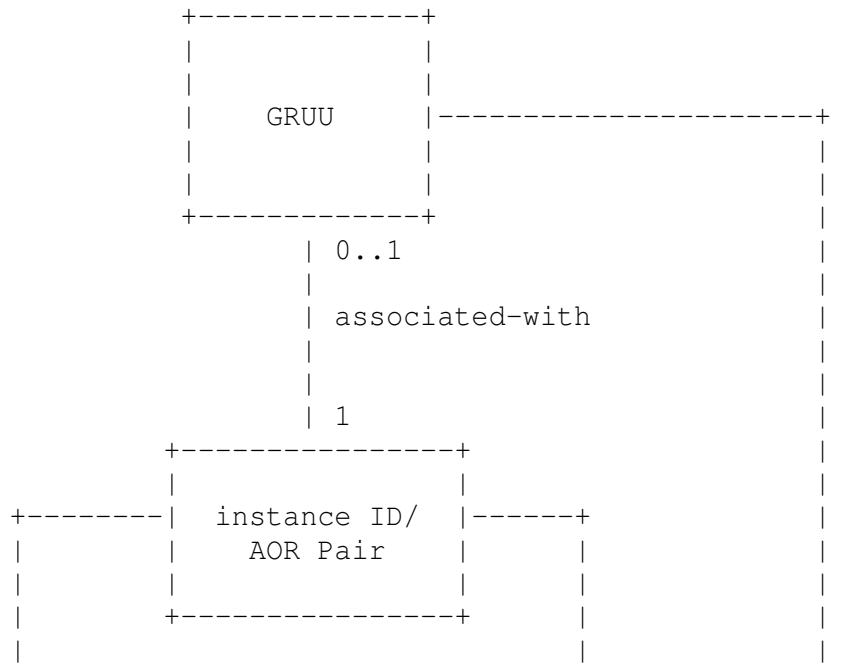
A UA can obtain a GRUU by generating a normal REGISTER request, as specified in RFC 3261 [1]. This request contains a Supported header field with the value "gruu", indicating to the registrar that the UA supports this extension. The UA includes a "sip.instance" media feature tag in the Contact header field of each Contact for which a GRUU is desired. This media feature tag contains a globally unique ID that identifies the UA instance. If the domain that the user is registering against also supports GRUU, the REGISTER responses will contain the "gruu" parameter in each Contact header field. This parameter contains a GRUU which the domain guarantees will route to that UA instace. That GRUU is guaranteed to remain valid for the duration of the registration. The GRUU is bound to the UA instace. Should the client change its Contact URI, but indicate that it represents the same instance ID, the server would provide the same GRUU. Furthermore, if the registration for the Contact expires, and the UA registers the Contact at a later time with the same instance identifier, the server would provide the same GRUU.

Since the GRUU is a URI like any other, it can be handed out by a UA by placing it in any header field which can contain a URI. A UA will normally place the GRUU into the Contact header field of dialog creating requests and responses it generates. However, it is important for the UA receiving the message to know whether the Contact URI is a GRUU or not. To make this determination, the UA looks for the presence of the Supported header field in the request or response. If it is present with a value of "gruu", it means that the Contact URI is a GRUU.

When a UA uses a GRUU, it has the option of adding the "grid" URI parameter to the GRUU. This parameter is opaque to the proxy server handling the domain. However, when the server maps the GRUU to the corresponding Contact URI, the server will copy the grid parameter into the Contact URI. As a result, when the UA receives the request, the Request URI will contain the grid parameter it placed in the corresponding GRUU.

6. Creation of a GRUU

A GRUU is a URI that is created and maintained by a server authoritative for the domain in which the GRUU resides. Independently of whether the GRUU is created as a result of a registration or some other means, a server MUST maintain certain information associated with the GRUU. This information, and its relationship with the GRUU, are modeled in Figure 2.



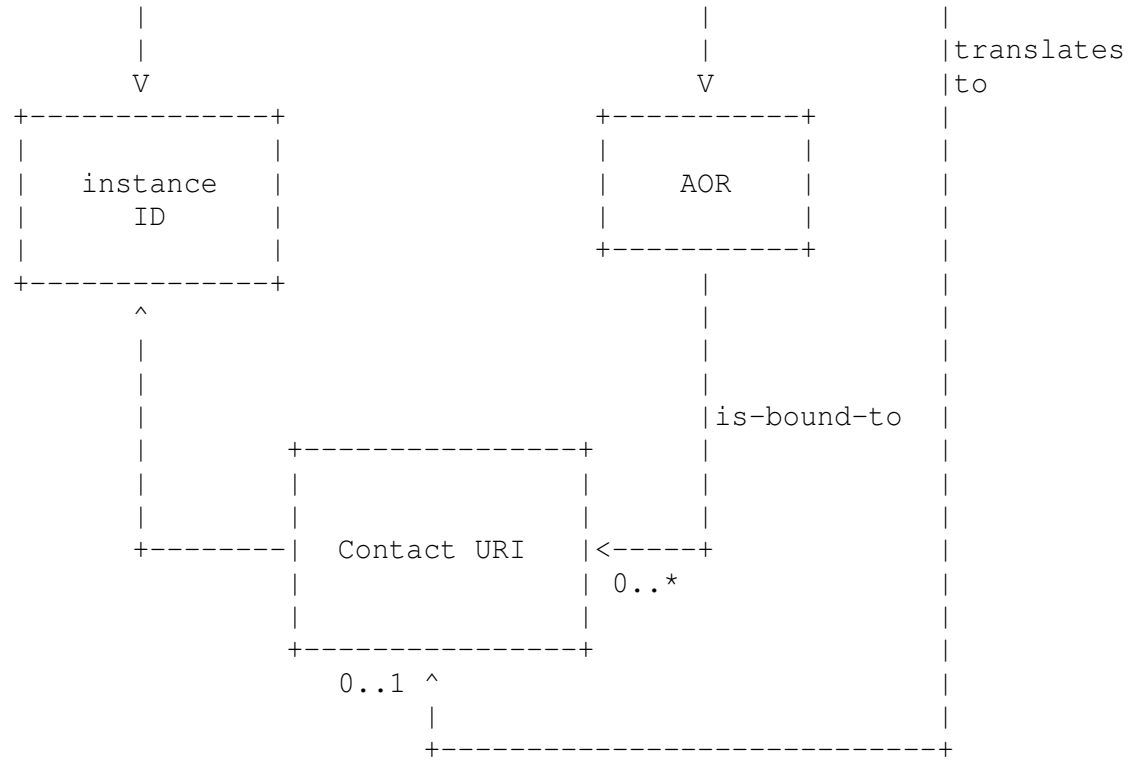


Figure 2

The instance ID plays a key role in this specification. It is an indentifier, represented by a URI, that uniquely identifies a SIP user agent amongst all other user agents with a Contact URI bound to an Address of Record (AOR). The instance ID allows a domain to create a GRUU that maps to the same UA instance, even if the Contact URI of that instance changes. Furthermore, the instance ID allows a domain to enforce the restriction that a specific UA instance can only be registered once against an AOR. When elements compliant to this specification compare two instance IDs for equality, the comparison is done using the equality rules for the scheme associated with that URI.

A GRUU is associated, in a one-to-one fashion, with the combination of an Address of Record (AOR) and instance ID. The GRUU is said to be associated with the combination, and the combination is associated with the GRUU. This combination is referred to as an instance ID/AOR pair. The instance ID/AOR pair serve to uniquely identify a user agent instance servicing a specific AOR. The AOR identifies a resource, such as a user or service within a domain, and the instance ID identifies a specific UA instance servicing requests for that resource.

It is important to understand that this uniqueness is over the instance ID/AOR pair, not just the instance ID. For example, if a user registered the Contact sip:ua@pc.example.com;+sip.instance="urn:foo:1", representing a device with instance ID urn:foo:1, to the AOR sip:user@example.com, and also registered the same Contact, representing the same instance ID - sip:ua@pc.example.com;+sip.instance="urn:foo:1" to a second AOR, say sip:boss@example.com, each of those UA instances would have a different GRUU, since they belong to different AORs.

A GRUU translates to zero or one Contact URIs. The Contact URI is a temporary URI that can be used to reach the instance ID/AOR pair. This URI can change due to changes in the IP address associated with the instance ID/AOR pair. If the instance ID associated with the GRUU is the instance ID of a Contact URI currently bound to the AOR associated with that GRUU, then the GRUU translates to that Contact URI. If, however, the instance ID associated with the GRUU is not an instance ID of a Contact URI currently bound to the AOR associated with the GRUU (possibly because there are no Contact URIs bound to the AOR), the GRUU maps to no Contact URI, and the GRUU is said to be invalid.

This specification does not mandate a particular mechanism for construction of the GRUU. Several example approaches are given in Appendix A. However, the GRUU MUST exhibit the following properties:

- The domain part of the URI is an IP address present on the public Internet, or, if it is a host name, exists in the global DNS and corresponds to an IP address present on the public Internet.
- When a request is sent to this URI, it routes to a proxy server in the same domain as that of the registrar.
- A proxy server in the domain can determine that the URI is a GRUU.
- When a proxy server in this domain receives a request sent to a URI that is a GRUU, that URI MUST be translated to the Contact URI currently bound to the AOR associated with that GRUU whose instance ID is the one associated with the GRUU.

Once an association from an instance ID/AOR to a GRUU is created, that mapping MUST remain in existence, and valid, as long as there exists any Contact bound to that AOR whose instance ID is that instance ID. If, through a de-registration or expiration, there is no longer any Contact bound to that AOR whose instance ID is that instance ID, the registrar MUST remove the mapping, and invalidate the GRUU. However, at any time in the future, should a Contact become bound to that same AOR, and that Contact is associated with the same instance ID, the domain SHOULD create the same GRUU that was previously associated with that instance ID/AOR pair. Indeed, this requirement would ideally be a MUST if it was achieveable, but even with the stateless algorithm described above, key rotation or server

failures may cause the GRUU associated with an instance ID/AOR pair to change. The value of associating the GRUU with an instance ID/AOR pair, as opposed to a Contact URI/AOR pair, is that the association can transcend changes in IP address. As a result, domains SHOULD make every effort possible to maintain the association for as long as possible.

7. Obtaining a GRUU

A GRUU can be obtained in many ways. This document defines two - through registrations, and through administrative operation.

7.1 Through Registrations

When a GRUU is associated with a user agent that comes and goes, and therefore registers to the network to bind itself to an AOR, a GRUU is provided to the user agent through SIP REGISTER messages.

7.1.1 User Agent Behavior

When a UA compliant to this specification generates a REGISTER request (initial or refresh), it MUST include the Supported header field in the request. The value of that header field MUST include "gruu" as one of the option tags. This alerts the registrar for the domain that the UA supports the GRUU mechanism.

Furthermore, for each Contact for which the UA desires to obtain a GRUU, the UA MUST include a "sip.instance" media feature tag as a UA characteristic [9]. As described in [9], this media feature tag will be encoded in the Contact header field as the "+sip.instance" Contact header field parameter. The value of this parameter MUST be a URI [7]. [9] defines equality rules for callee capabilities parameters, and according to that specification, the "sip.instance" media feature tag will be compared by case sensitive string comparison. Those equality rules apply only to the generic usages defined there and in the caller preferences specification [16]. When the instance ID is used in this specification, it is effectively "extracted" from the value in the "sip.instance" media feature tag, and thus equality comparisons are performed using the rules for URI equality specific to the scheme in the URI.

It is RECOMMENDED that the URI be a Uniform Resource Name (URN) [8]. This specification makes no normative recommendation on the specific URI or URN that is to be used. However, the URI MUST be selected such that the instance can be certain that no other instance registering against the same AOR would choose the same URI value. Usage of a URN is RECOMMENDED since it provides a persistent and unique name for the UA instance, allowing it to obtain the same GRUU

over time. It also provides an easy way to guarantee uniquess within the AOR. However, this specification does not require a long-lived and persistent instance identifier to properly function, and in some cases, there may be cause to use an identifier with weaker temporal persistence.

One URN that readily meets the requirements of this specification is the UUID URN [20], which allows for non-centralized computation of a URN based on time, unique names (such as a MAC address) or a random number generator. An example of a URN that would not meet the requirements of this specification is the national bibliographic number [13]. Since there is no clear relationship between an SIP UA instance and a URN in this namespace, there is no way a selection of a value can be performed that guarantees that another UA instance doesn't choose the same value.

Besides the presence of the "gruu" option tag in the Supported header field and the "+sip.instance" Contact header field parameter, the REGISTER request is constructed identically to the case where this extension was not understood. Specifically, the Contact URI in the REGISTER request SHOULD NOT contain the gruu Contact header field parameter. Any such parameters are ignored by the registrar, as the UA cannot propose a GRUU for usage with the Contact URI.

If a UA wishes to guarantee that the request is not processed unless the domain supports and uses this extension, it MAY include a Require header field in the request with a value that contains the "gruu" option tag.

If the response is a 2xx, each Contact header field that contained the "+sip.instance" Contact header field parameter may also contain a "gruu" parameter. This parameter contains a SIP URI that represents a GRUU corresponding to the UA instance that registered the contact. Any requests sent to the GRUU URI will be routed by the domain to the Contact URI currently bound to that instance ID. The GRUU will not normally change in subsequent 2xx responses to REGISTER. Indeed, even if the UA lets the contact expire, when it re-registers it at any later time, the registrar will normally provide the same GRUU for the same address-of-record and instance ID. However, this property cannot be completely guaranteed, as network failures may make it impossible to provide an identifier that persists for all time. As a result, a UA MUST be prepared to receive a different GRUU in a subsequent registration response.

A non-2xx response to the REGISTER request has no impact on any existing GRUU previously provided to the UA. Specifically, if a previously successful REGISTER request provided the UA with a GRUU, a subsequent failed request does not remove, delete, or otherwise

invalidate the GRUU.

If the response to the REGISTER request was a 425, it means that one of the Contact URI in the REGISTER request contained an instance ID that was already associated with a different registered Contact. It is up to the client to resolve this conflict. The conflict normally arises when a client registers a Contact with its instance ID, crashes, and reboots. After reboot, it obtains a new IP address, and attempts to register a Contact for that address, containing the same instance ID. In such a case, the proper course of action is to remove the old registration. To do that, the client can send a REGISTER request with no Contacts. The 200 OK contains the list of currently registered Contacts, including their instance IDs. The client can find the existing contact that matches its instance ID, and then send a new REGISTER request. This request would include the old Contact, with the instance ID, and an expires value of 0. Then, the client can retry its failed registration.

7.1.2 Registrar Behavior

A registrar MAY create a GRUU for a particular instance ID/AOR pair at any time. Of course, if a UA requests a GRUU in a registration, and the registrar has not yet created one, it will need to do so in order to respond to the registration request. However, the registrar can create the GRUU in advance of any request from a UA.

When a registrar compliant to this specification receives a REGISTER request, it checks for the presence of the Require header field in the request. If present, and if it contains the "gruu" option tag, the registrar MUST follow the procedures in the remainder of this section (that is, the procedures which result in the creation of new GRUUs for Contacts indicating an instance ID, and the listing of GRUUs in the REGISTER response). If not present, but a Supported header field was present with the "gruu" option tag, the registrar SHOULD follow the procedures in the remainder of this section. If the Supported header field was not present, or it if was present but did not contain the value "gruu", the registrar SHOULD NOT follow the procedures in the remainder of this section.

As the registrar is processing the Contacts in the REGISTER request according to the procedures of step 7 in Section 10.3 of RFC 3261, the registrar additionally checks whether each contact contains a "+sip.instance" header field parameter. If it does, the registrar takes the value of that parameter as an instance ID. The registrar checks to see if there is any other contact bound to the same AOR with the same instance ID (recall that equality is computed using URI equality for the scheme in question). If there is, this is an error condition. Only a single Contact URI at a time can be registered for

each instance ID. As a result, the registrar MUST reject the request with a 425 (Instance Conflict) error response. This response code informs the client that its registration failed because the instance ID provided in the request is already registered to a different Contact. It is up to the client to decide how to proceed.

If there is no other contact bound to the same AOR with the same instance ID, the server allocates and/or creates a GRUU for that instance ID/AOR pair according to the procedures of Section 6. If the contact contained a "gruu" Contact header field parameter, it MUST be ignored by the registrar. A UA cannot suggest or otherwise provide a GRUU to the registrar. In addition to storing the contact URI, the server MUST store the instance ID.

When generating the 200 (OK) response to the REGISTER request, the procedures of step 8 of Section 10.3 of RFC 3261 are followed. Furthermore, for each Contact header field value placed in the response, if the registrar has stored an instance ID associated with that contact URI, the server MUST add a "gruu" Contact header field parameter. This parameter contains the instance ID for the user agent. The value of the gruu parameter is a quoted string containing the URI that is the GRUU for the associated instance ID/AOR pair.

Note that handling of a REGISTER request containing a Contact header field with value "*" and an expiration of 0 still retains the meaning defined in RFC 3261 – all Contacts, not just ones with a specific instance ID, are deleted.

Inclusion of a GRUU in the "gruu" Contact header field parameter of a REGISTER response is separate from the computation and storage of the GRUU. It is possible that the registrar has computed a GRUU for one UA, but a different UA that queries for the current set of registrations doesn’t understand GRUU. In that case, the REGISTER response sent to that second UA would not contain the "gruu" Contact header field parameter, even though the UA has a GRUU for that Contact.

7.2 Administratively

Administrative creation of GRUUs is useful when a UA instance is a network server that is always available, and therefore doesn’t register to the network. Examples of such servers are voicemail servers, application servers, and gateways.

There are no protocol operations required to administratively create a GRUU. The proxy serving the domain is configured with the GRUU, and with the Contact URI it should be translated to. It is not strictly necessary to also configure the instance ID and AOR, since

the translation can be done directly. However, they serve as a useful tool for determining which resource and UA instance the GRUU is supposed to map to.

In addition to configuring the GRUU and its associated Contact URI in the proxy serving the domain, the GRUU will also need to be configured into the UA instance associated with the GRUU.

8. Using the GRUU

8.1 Sending a Message Containing a GRUU

A UA first obtains a GRUU using the procedures of Section 7, or by other means outside the scope of this specification.

A UA can use the GRUU in the same way it would use any other SIP URI. However, a UA compliant to this specification MUST use a GRUU when populating the Contact header field of dialog-creating requests and responses. This includes the INVITE request and its 2xx response, the SUBSCRIBE [4] request, its 2xx response, the NOTIFY request, and the REFER [5] request and its 2xx response. Similarly, in those requests and responses where the GRUU is used in the Contact header field, the UA MUST include a Supported header field that contains the option tag "gruu". However, it is not necessary for a UA to know whether or not its peer in the dialog uses a GRUU before inserting one into the Contact header field.

When placing a GRUU into the Contact header field of a request or response, a UA MAY add the "grid" URI parameter to the GRUU. This parameter MAY take on any value permitted by the grammar for the parameter. Note that there are no limitations on the size of this parameter. When a UA sends a request to the GRUU, the proxy for the domain that owns the GRUU will translate the GRUU in the Request-URI, replacing it with the corresponding Contact URI. However, it will retain the "grid" parameter when this translation is performed. As a result, when the UA receives the request, the Request-URI will contain the "grid" created by the UA. This allows the UA to effectively manufacture an infinite supply of GRUU, each of which differs by the value of the "grid" parameter. When a UA receives a request that was sent to the GRUU, it will be able to tell which GRUU was invoked by the "grid" parameter.

An implication of this behavior is that all mid-dialog requests will be routed through intermediate proxies. There will never be direct, UA to UA signaling. It is anticipated that this limitation will be addressed in future specifications.

Once a UA knows that the Contact URI provided by its peer is a GRUU,

it can use it in any application or SIP extension which requires a globally routable URI to operate. One such example is assisted call transfer.

8.2 Sending a Message to a GRUU

There is no new behavior associated with sending a request to a GRUU. A GRUU is a URI like any other. When a UA receives a request or response, it will know that the Contact header field contained a GRUU if the request or response had a Supported header field that included the value "gruu". The UA can take the GRUU, and send a request to it, and then be sure that it is delivered to the UA instance which sent the request or response.

Since the instance ID is a callee capabilities parameter, a UA might be tempted to send a request to the AOR of a user, and include an Accept-Contact header field [16] which indicates a preference for routing the request to a UA with a specific instance ID. Although this would appear to have the same effect as sending a request to the GRUU, it does not. The caller preferences expressed in the Accept-Contact header field are just preferences, and do not work with the some reliability as GRUU. However, this specification does not forbid a client from attempting such a request, as there may be cases where the desired operation truly is a preferential routing request.

8.3 Receiving a Request Sent to a GRUU

When a UAS receives a request sent to its GRUU, the incoming request URI will be equal to the Contact URI that was registered (through REGISTER or some other action) by that UA instance. If the user agent had previously handed out its GRUU with a grid parameter, the incoming request URI may contain that parameter. This indicates to the UAS that the request is being received as a result of a request sent by the UAC to that GRUU/grid combination. This specification makes no normative statements about when to use a grid parameter, or what to do when receiving a request made to a GRUU/grid combination. Generally, any differing behaviors are a matter of local policy.

It is important to note that, when a user agent receives a request, and the request URI does not have a grid parameter, the user agent cannot tell whether the request was sent to the AOR or to the GRUU. As such, the UAS will process such requests identically. If a user agent needs to differentiate its behavior based on these cases, it will need to use a grid parameter.

8.4 Proxy Behavior

When a proxy server receives a request, and the proxy owns the domain in the Request URI, and the proxy is supposed to access a Location Service in order to compute request targets (as specified in Section 16.5 of RFC 3261 [1]), the proxy MUST check if the Request URI is a GRUU created by that domain.

If the URI is a GRUU, the proxy MUST determine if there is still a Contact URI bound to AOR associated with the GRUU, whose instance ID is the instance ID associated with the GRUU. If that AOR no longer has any contacts bound to it, or if it does have contacts bound to it, but none of them have an instance ID equal to the instance ID associated with the GRUU, the proxy MUST generate a 480 (Temorarily Unavailable) response to the request. If, however, the proxy does not recognize the GRUU as one it had constructed previously for the domain, the proxy MUST generate a 404 (Not Found) response to the request.

Otherwise, the proxy MUST populate the target set with a single URI. This URI MUST be equal to the Contact URI that is translated from the GRUU. Furthermore, if the GRUU contained a "grid" URI parameter, the URI in the target set MUST also contain the same parameter with the same value.

A proxy MAY apply other processing to the request, such as execution of called party features. In particular, it is RECOMMENDED that non-routing called party features, such as call logging and screening, that are associated with the AOR are also applied to requests for all GRUUs associated with that AOR.

In many cases, a proxy will record-route an initial INVITE request, and the user agents will insert a GRUU into the Contact header field. When this happens, a mid-dialog request will arrive at the proxy with a Route header field that was inserted by the proxy, and a Request-URI that represents a GRUU. Proxies follow normal processing in this case; they will strip the Route header field, and then process the Request URI as described above.

The procedures of RFC 3261 are then followed to proxy the request. The request SHOULD NOT be redirected in this case. In many instances, a GRUU is used by a UA in order to assist in the traversal of NATs and firewalls, and a redirection may prevent such a case from working.

9. 425 (Instance Conflict) Response Code

This specification defines a new response code for SIP. The response

code is 425, and it has a default reason phrase of "Instance Conflict". This response code is valid only for REGISTER responses. It informs the UA that its registration failed because the instance ID provided in the request is already registered to a different Contact.

10. Grammar

This specification defines two new Contact header field parameters, gruu and +sip.instance, and a new URI parameter, grid. The grammar for string-value is obtained from [9], and the grammar for uric is defined in RFC 2396 [7].

```
contact-params    =  c-p-q / c-p-expires / c-p-gruu / cp-instance
                      / contact-extension
c-p-gruu          =  "gruu" EQUAL DQUOTE SIP-URI DQUOTE
cp-instance       =  "+sip.instance" EQUAL LDQUOT instance-val RDQUOT
uri-parameter     =  transport-param / user-param / method-param
                     / ttl-param / maddr-param / lr-param / grid-param
                     / other-param
grid-param        = "grid=" pvalue        ; defined in RFC3261
instance-val      = uric ; defined in RFC 2396
```

11. Requirements

This specification was created in order to meet the following requirements:

REQ 1: When a UA invokes a GRUU, it MUST cause the request to be routed to the specific UA instance to which the GRUU refers.

REQ 2: It MUST be possible for a GRUU to be invoked from anywhere on the Internet, and still cause the request to be routed appropriately. That is, a GRUU MUST NOT be restricted to use within a specific addressing realm.

REQ 3: It MUST be possible for a GRUU to be constructed without requiring the network to store additional state.

REQ 4: It MUST be possible for a UA to obtain a multiplicity of GRUUs, each one of which routes to that UA instance. This is needed to support ad-hoc conferencing, for example, where a a UA instance needs a different URI for each conference it is hosting.

REQ 5: When a UA receives a request sent to a GRUU, it MUST be possible for the UA to know the GRUU which was used to invoke the request. This is necessary as a consequence of requirement 4.

REQ 6: It MUST be possible for a UA to add opaque content to a GRUU, which is not interpreted or altered by the network, and used only by the UA instance to whom the GRUU refers. This provides a basic cookie type of functionality, allowing a UA to build a GRUU with

state embedded within it.

REQ 7: It MUST be possible for a proxy to execute services and features on behalf of a UA instace represented by a GRUU. As an example, if a user has call blocking features, a proxy may want to apply those call blocking features to calls made to the GRUU in addition to calls made to the user's AOR.

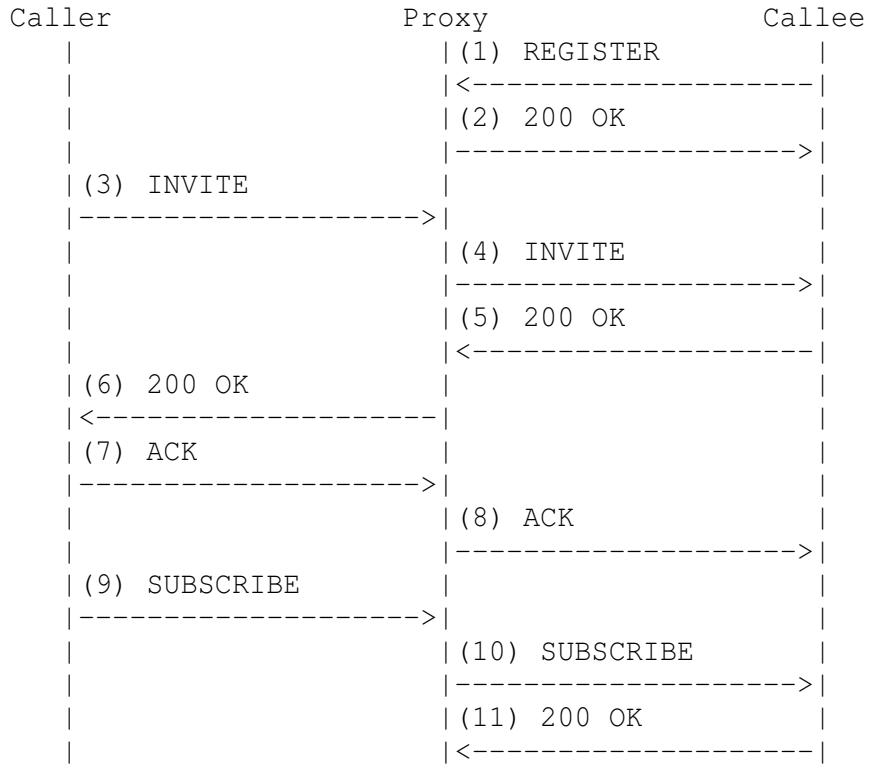
REQ 8: It MUST be possible for a UA in a dialog to inform its peer of its GRUU, and for the peer to know that the URI represents a GRUU. This is needed for the conferencing and dialog reuse applications of GRUUs, where the URIs are transferred within a dialog.

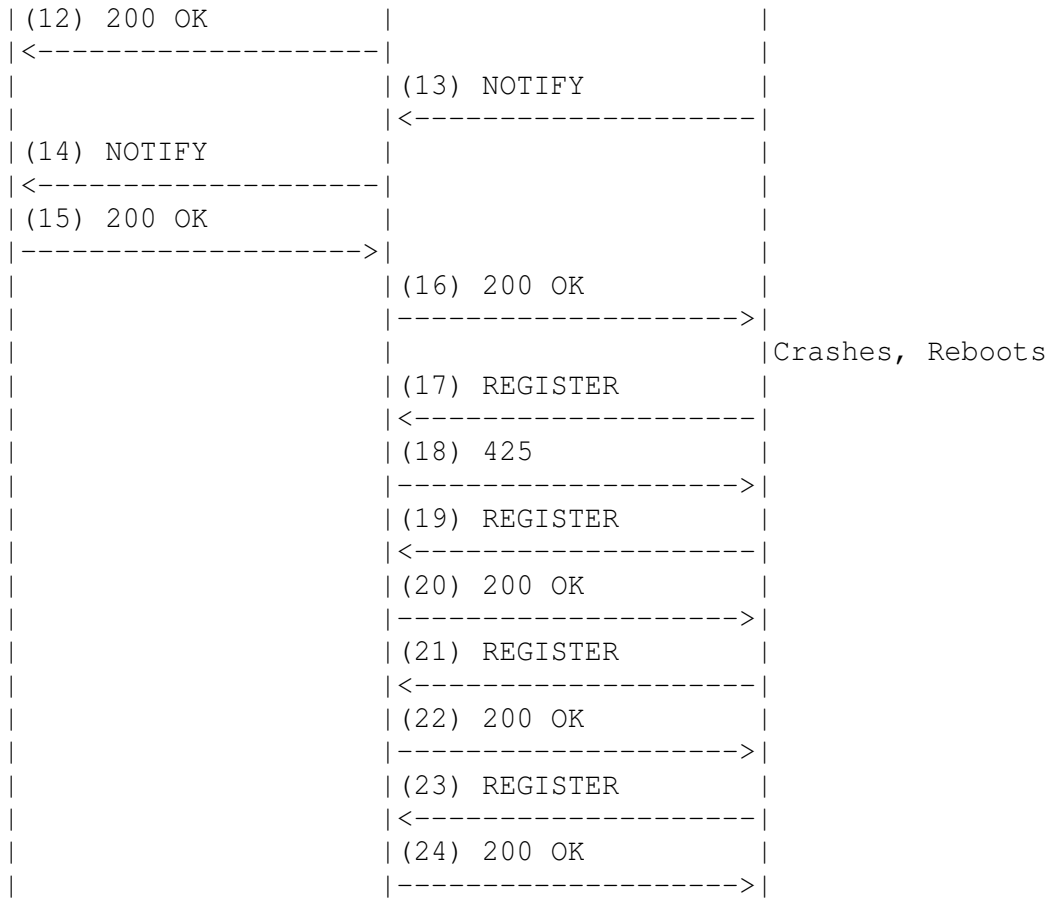
REQ 9: When transferring a GRUU per requirement 8, it MUST be possible for the UA receiving the GRUU to be assured of its integrity and authenticity.

REQ 10: It MUST be possible for a server, authoritative for a domain, to construct a GRUU which routes to a UA instace bound to an AOR in that domain. In other words, the proxy can construct a GRUU too. This is needed for the presence application.

12. Example Call Flow

The following call flow shows a basic registration and call setup, followed by a subscription directed to the GRUU. It then shows a failure of the callee, followed by a re-registration.





The Callee supports the GRUU extension. As such, its REGISTER (1) looks like:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.1;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=a73kszlfl
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: 1j9FpLxk3uxtm8tn@192.0.2.1
CSeq: 1 REGISTER
Contact: <sip:callee@192.0.2.1>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
Content-Length: 0
```

The REGISTER response would look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.1;branch=z9hG4bKnashds7
From: Callee <sip:callee@example.com>;tag=a73kszlfl
To: Callee <sip:callee@example.com> ;tag=b88sn
Call-ID: 1j9FpLxk3uxtm8tn@192.0.2.1
CSeq: 1 REGISTER
Contact: <sip:callee@192.0.2.1>
  ;gruu="sip:hha9s8d=-999a@example.com"
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
  ;expires=3600
Content-Length: 0
```

Note how the Contact header field in the REGISTER response contains the gruu parameter with the URI sip:hha9s8d=-999a@example.com. This represents a GRUU that translates to the Contact URI sip:callee@192.0.2.1.

The INVITE from the caller is a normal SIP INVITE. The 200 OK generated by the callee, however, now contains a GRUU in the Contact header field. The UA has also chosen to include a grid URI parameter into the GRUU.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bKnaa8
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK99a
From: Caller <sip:caller@example.com>;tag=n88ah
To: Callee <sip:callee@example.com> ;tag=a0z8
Call-ID: 1j9FpLxk3uxtma7@host.example.com
CSeq: 1 INVITE
Supported: gruu
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:hha9s8d=-999a@example.com;grid=99a>
Content-Length: --
Content-Type: application/sdp

[SDP Not shown]
```

At some point later in the call, the caller decides to subscribe to the dialog event package [15] at that specific UA. To do that, it generates a SUBSCRIBE request (message 9), but directs it towards the GRUU contained in the Contact header field.

```
SUBSCRIBE sip:hha9s8d=-999a@example.com;grid=99a SIP/2.0
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:caller@example.com>;tag=kkaz-
To: Callee <sip:callee@example.com>
Call-ID: faif9a@host.example.com
CSeq: 2 SUBSCRIBE
Supported: gruu
Event: dialog
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:bad998asd8asd0000a0@example.com>
Content-Length: 0
```

In this example, the caller itself supports the GRUU extension, and is using its own GRUU to populate the Contact header field of the SUBSCRIBE.

This request is routed to the proxy, which proceeds to perform a location lookup on the request URI. It is translated into the Contact URI of that GRUU, and then proxied there (message 10 below). Note how the grid parameter is maintained.

```
SUBSCRIBE sip:callee@192.0.2.1;grid=99a SIP/2.0
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bK9555
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:caller@example.com>;tag=kkaz-
To: Callee <sip:callee@example.com>
Call-ID: faif9a@host.example.com
CSeq: 2 SUBSCRIBE
Supported: gruu
Event: dialog
Allow: INVITE, OPTIONS, CANCEL, BYE, ACK
Contact: <sip:bad998asd8asd0000a0@example.com>
Content-Length: 0
```

At some point after message 16 is received, the callee’s machine crashes and recovers. It obtains a new IP address, 192.0.2.2. Unaware that it had previously had an active registration, it creates a new one (message 17 below). Notice how the instance ID remains the same, as it persists across reboot cycles:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbba
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=ha8d777f0
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: hf8asxzff8s7f@192.0.2.2
CSeq: 1 REGISTER
Contact: <sip:callee@192.0.2.2>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
Content-Length: 0
```

The registrar notices that a different contact, sip:callee@192.0.2.1, is already associated with the same instance ID. Thus, it rejects the request in message 18, below:

```
SIP/2.0 425 Instance Conflict
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbba
From: Callee <sip:callee@example.com>;tag=ha8d777f0
To: Callee <sip:callee@example.com>;tag=776554
Call-ID: hf8asxzff8s7f@192.0.2.2
CSeq: 1 REGISTER
```

Next, the client formulates a new REGISTER request, to query for the existing set of registrations (message 19, below):

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbb
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=ha8d777f1
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 2 REGISTER
```

This generates a 200 (OK) response (message 20, below) that includes the existing contact:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbb
From: Callee <sip:callee@example.com>;tag=ha8d777f1
To: Callee <sip:callee@example.com>;tag=8asd7d666
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 2 REGISTER
Contact: <sip:callee@192.0.2.1>
```

```
  ;gruu="sip:hha9s8d=-999a@example.com"
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
  ;expires=2000
```

The client realizes that a different IP address is registered with the same instance ID. Since the client knows that its instance ID is globally unique, it deletes that registration (message 21, below):

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbc
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=ha8d777f2
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 3 REGISTER
Contact: <sip:callee@192.0.2.1>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
  ;expires=0
```

This deletes the contact, as indicated by the lack of of the Contact header field in the resulting 200 OK (message 22, below):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbc
From: Callee <sip:callee@example.com>;tag=ha8d777f2
To: Callee <sip:callee@example.com>;tag=7asdnj7d6f
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 3 REGISTER
```

Finally, the client can retry its original registration (message 23, below):

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbd
Max-Forwards: 70
From: Callee <sip:callee@example.com>;tag=ha8d777f3
Supported: gruu
To: Callee <sip:callee@example.com>
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 4 REGISTER
Contact: <sip:callee@192.0.2.2>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
```

This time, the registration succeeds, and the client is registered.

The response, message 24, is shown below:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.2;branch=z9hG4bKnasbbd
From: Callee <sip:callee@example.com>;tag=ha8d777f3
To: Callee <sip:callee@example.com>;tag=asd7salll
Call-ID: hf8asxzff8s7g@192.0.2.2
CSeq: 4 REGISTER
Contact: <sip:callee@192.0.2.2>
  ;+sip.instance="<urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>"
  ;expires=3600
```

13. Security Considerations

GRUUs do not provide a complete or reliable solution for privacy. In particular, since the GRUU does not change during the lifetime of a registration, an attacker could correlate two calls as coming from the same source, which in and of itself reveals information about the caller. Furthermore, GRUUs do not address other aspects of privacy, such as the addresses used for media transport. For a discussion of how privacy services are provided in SIP, see RFC 3323 [12].

It is important for a UA to be assured of the integrity of a GRUU when it is given one in a REGISTER response. If the GRUU is tampered with by an attacker, the result could be denial of service to the UA. As a result, it is RECOMMENDED that a UA use the SIPS URI scheme when registering.

14. IANA Considerations

This specification defines a new Contact header field parameter, a new SIP response code, a SIP URI parameter, a media feature tag and a SIP option tag.

14.1 Header Field Parameter

This specification defines a new header field parameter, as per the registry created by [10]. The required information is as follows:
Header field in which the parameter can appear: Contact
Name of the Parameter gruu
RFC Reference RFC XXXX [[NOTE TO IANA: Please replace XXXX with the RFC number of this specification.]]

14.2 Response Code

This specification defines the new SIP response code, 425, per the

guidelines in Section 27.4 of RFC 3261.
RFC Number: This specification, RFC XXXX [[NOTE to IANA: Please replace XXXX with the RFC number for this specification.]].
Response Code Number: 425
Default Reason Phrase: Instance Conflict

14.3 URI Parameter

This specification defines a new SIP URI parameter, as per the registry created by [11].
Name of the Parameter grid
RFC Reference RFC XXXX [[NOTE TO IANA: Please replace XXXX with the RFC number of this specification.]]

14.4 Media Feature Tag

This section registers a new media feature tag, per the procedures defined in RFC 2506 [6]. The tag is placed into the sip tree, which is defined in [9].
Media feature tag name: sip.instance
ASN.1 Identifier: New assignment by IANA.
Summary of the media feature indicated by this tag: This feature tag contains a string containing a URI, and ideally a URN, that indicates a unique identifier associated with the UA instance registering the Contact.
Values appropriate for use with this feature tag: String.
The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application, for describing the capabilities of a device, such as a phone or PDA.
Examples of typical use: Routing a call to a specific device.
Related standards or documents: RFC XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]
Security Considerations: This media feature tag can be used in ways which affect application behaviors. For example, the SIP caller preferences extension [16] allows for call routing decisions to be based on the values of these parameters. Therefore, if an attacker can modify the values of this tag, they may be able to affect the behavior of applications. As a result of this, applications which utilize this media feature tag SHOULD provide a means for ensuring its integrity. Similarly, this feature tag should only be trusted as valid when it comes from the user or user agent described by the tag. As a result, protocols for conveying this feature tag SHOULD provide a mechanism for guaranteeing authenticity.

14.5 SIP Option Tag

This specification registers a new SIP option tag, as per the guidelines in Section 27.1 of RFC 3261.
Name: gruu
Description: This option tag is used to identify the Globally Routable User Agent URI (GRUU) extension. When used in a Supported header, it indicates that a User Agent understands the extension, and has included a GRUU in the Contact header field of its dialog initiating requests and responses. When used in a Require header field of a REGISTER request, it indicates that the registrar should assign a GRUU to the Contact URI.

15. Acknowledgements

The author would like to thank Rohan Mahy, Paul Kyzivat, Alan Johnston, and Cullen Jennings for their contributions to this work.

16. References

16.1 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., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

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

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

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

[6] Holtman, K., Mutz, A. and T. Hardie, "Media Feature Tag Registration Procedure", BCP 31, RFC 2506, March 1999.

[7] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

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

[9] Rosenberg, J., "Indicating User Agent Capabilities in the

Session Initiation Protocol (SIP)", draft-ietf-sip-callee-caps-03 (work in progress), January 2004.

[10] 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.

[11] Camarillo, G., "The Internet Assigned Number Authority (IANA) Universal Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", draft-ietf-sip-uri-parameter-reg-02 (work in progress), June 2004.

16.2 Informative References

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

[13] Hakala, J., "Using National Bibliography Numbers as Uniform Resource Names", RFC 3188, October 2001.

[14] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-01 (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] 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.

[17] Sugano, H. and S. Fujimoto, "Presence Information Data Format (PIDF)", draft-ietf-impp-cpim-pidf-08 (work in progress), May 2003.

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

[19] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", draft-ietf-simple-presence-10 (work in progress), January 2003.

[20] Mealling, M., "A UUID URN Namespace",
draft-mealling-uuid-urn-03 (work in progress), 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

Appendix A. Example GRUU Construction Algorithms

The mechanism for constructing a GRUU is not subject to specification. This appendix provides two examples that can be used by a registar. Others are, of course, permitted, as long as they meet the constraints defined for a GRUU.

A.1 Encrypted Instance ID and AOR

In many cases, it will be desirable to construct the GRUU in such a way that it will not be possible, based on inspection of the URI, to determine the Contact URI that the GRUU translates to. It may also be desirable to construct it so that it will not be possible to determine the instance ID/AOR pair associated with the GRUU. Whether or not a GRUU should be constructed with this property is a local policy decision.

With these rules, it is possible to construct a GRUU without requiring the maintenance of any additional state. To do that, the URI would be constructed in the following fashion:

```
user-part = "GRUU" + BASE64(E(K, (salt + " " + AOR + " " +
instance ID)))
```

Where E(K,X) represents a suitable encryption function (such as AES with 128 bit keys) with key K applied to data block X, and the "+" operator implies concatenation. The single space (" ") between components is used as a delimeter, so that the components can easily be extracted after decryption. Salt represents a random string that prevents a client from obtaining pairs of known plaintext and ciphertext. A good choice would be at least 128 bits of randomness in the salt.

The benefit of this mechanism is that a server need not store additional information on mapping a GRUU to its corresponding Contact URI. The user part of the GRUU contains the instance ID and AOR. Assuming that the domain stores registrations in a database indexed by the AOR, the proxy processing the GRUU would look up the AOR, extract the currently registered Contacts, and find the one matching the instance ID encoded in the request URI. The Contact URI whose instance ID is that instance ID is then used as the translated version of the URI. Encryption is needed to prevent attacks whereby the server is sent requests with faked GRUU, causing the server to direct requests to any named URI. Even with encryption, the proxy should validate the user part after decryption. In particular, the AOR should be one managed by the proxy in that domain. Should a UA send a request with a fake GRUU, the proxy would decrypt and then discard it because there would be no URI or an invalid URI inside.

While this approach has many benefits, it has the drawback of producing fairly long GRUUs. The approach in the following section produces smaller results, at the cost of additional structures in the database.

A.2 Hashed Indices

As an alternative approach, the server can construct the GRUU by computing a cryptographic hash of the AOR and instance ID, taking 64 bits of the result, and placing a string representation of those 64 bits into the user part of the URI.

When a GRUU is created through registration or administrative action, the server computes this hash and stores the hash in the database. This hash acts the primary key, with the columns of the table providing the instance ID, AOR and Contact. When the registration is deleted, the corresponding row from the table is removed. When a request arrives to a proxy, the user part of the URI is looked up in the database, and the Contact, AOR and instance ID can be extracted.

This approach produces GRUUs of relatively short length. However, it requires additional structures to be created and stored in a database that would be used by the registrar (at least, new structures are needed for efficient operation). However, it does not require the registrar to store anything for longer than the duration of the registration.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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

INTERNET-DRAFT
Document: draft-ietf-sip-history-info-03.txt
Category: Standards Track

M. Barnes
Editor
Nortel Networks

Expires: January 8, 2005

July 8, 2004

An Extension to the Session Initiation Protocol for Request History Information

Status of this Memo

Copyright Notice

Abstract

This draft defines a standard mechanism for capturing the history information associated with a SIP request. This capability enables many enhanced services by providing the information as to how and why a call arrives at a specific application or user. This draft defines a new optional SIP header, History-Info, for capturing the history information in requests. A new option tag, Histinfo, to be included in the Supported header, is defined to allow UAs to indicate whether the History-Info should be returned in responses to a request which has captured the history information. A new priv-value, history, is

added to the Privacy header to allow for privacy handling of the History-Info header.

Table of Contents

Overview

Many services that SIP is anticipated to support require the ability to determine why and how the call arrived at a specific application. Examples of such services include (but are not limited to) sessions initiated to call centers via "click to talk" SIP URLs on a web page, "call history/logging" style services within intelligent "call management" software for SIP UAs and calls to voicemail servers and call centers. While SIP implicitly provides the redirect/retarget capabilities that enable calls to be routed to chosen applications, there is currently no standard mechanism within SIP for communicating the history of such a request. This "request history" information allows the receiving application to determine hints about how and why the call arrived at the application/user. This draft defines a new SIP header, History-Info, to provide a standard mechanism for capturing the request history information to enable a wide variety of

services for networks and end users. The History-Info header provides a building block for development of new services.

Section 1 provides additional background motivation for the Request History capability. Section 2 identifies the requirements for a solution, with Section 3 providing an overall description of the solution.

Section 4 provides the details of the additions to the SIP protocol. Example uses of the new header are included in Section 4.5, with additional scenarios included in the Appendix. It is anticipated that these would be moved and progressed in a general Service examples draft such as [SIPSVCEX] or individual informational drafts describing these specific services, since the History-Info header is just one of the building blocks for implementing these services. Individual drafts would be particularly useful for documenting services for which there are multiple solutions, as it is not the intent, nor is it within the scope, of this draft to prescribe a complete solution for any of these applications.

Section 5 summarizes the application considerations identified in the previous sections. Section 6 summarizes the security solution.

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 [RFC2119].

In order to provide a cross reference of the solution description to the requirements without reiterating the entirety of the requirements inline, the requirements are referenced as [REQNAME-req] following the text or paragraph which explicitly satisfies the requirement.

1. Background: Why define a Generic "Request History" capability?

SIP implicitly provides redirect/retarget capabilities that enable calls to be routed to specific applications as defined in [RFC3261]. The term retarget will be used henceforth in this draft to refer to the process of a Proxy Server/UAC changing a URI in a request and thus changing the target of the request. This term is chosen to avoid associating this request history only with the specific SIP Redirect Server capability that provides for a response to be sent back to a UAC requesting that the UAC should retarget the original request to an alternate URI. The rules for determining request targets as described in section 16.5 of [RFC3261] are consistent with the use of the retarget term in this draft.

The motivation for the request history is that in the process of retargeting old routing information can be forever lost. This lost information may be important history that allows elements to which the call is retargeted to process the call in a locally defined, application specific manner. The proposal in this draft is to provide a mechanism for transporting the request history. It is not proposing any application specific behavior for a Proxy or UA upon receipt of the information. Indeed, such behavior should be a local decision for the recipient application.

Current network applications provide the ability for elements involved with the call to exchange additional information relating to how and why the call was routed to a particular destination. The following are examples of such applications:

1. Web "referral" applications, whereby an application residing within a web server determines that a visitor to a website has arrived at the site via an "associate" site which will receive some "referral" commission for generating this traffic,

2. Email forwarding whereby the forwarded-to user obtains a "history" of who sent the email to whom and at what time

3. Traditional telephony services such as Voicemail, call-center "automatic call distribution", and "follow-me" style services.

Several of the aforementioned applications currently define application specific mechanisms through which it is possible to obtain the necessary history information.

In addition, request history information could be used to enhance basic SIP functionality by providing the following:

4. Some diagnostic information for debugging SIP requests.

5. A stronger security solution for SIP. A side effect is that each proxy which captures the "request history" information in a secure manner provides an additional means (without requiring signed keys) for the original requestor to be assured that the request was properly retargeted.

2. "Request History" Requirements

The following list constitutes a set of requirements for a "Request History" capability.

1) CAPABILITY-req: The "Request History" capability provides a capability to inform proxies and UAs involved in processing a request about the history/progress of that request. While this is inherently provided when the retarget is in response to a SIP redirect, it is deemed useful for non-redirect retargeting scenarios, as well.

2) OPTIONALITY-req: The "Request History" information is optional.

2.1) In many cases, it is anticipated that whether the history is added to the Request would be a local policy decision enforced by the specific application, thus no specific protocol element is needed.

2.2) Due to the capability being "optional" from the SIP protocol perspective, the impact to an application of not having the "Request History" must be described. Applicability guidelines to be addressed by applications using this capability must be provided as part of the solution to these requirements.

3) GENERATION-req: "Request History" information is generated when the request is retargeted.

3.1) In some scenarios, it might be possible for more than one instance of retargeting to occur within the same Proxy. A proxy should also generate Request History information for the 'internal retargeting'.

3.2) An entity (UA or proxy) retargeting in response to a redirect or REFER should include any Request History information from the redirect/REFER in the new request.

4) ISSUER-req: "Request History" information can be generated by a UA or proxy. It can be passed in both requests and responses.

5) CONTENT-req: The "Request History" information for each occurrence of retargeting, shall include the following:

5.1) The new URI or address to which the request is in the process of being retargeted,

5.2) The URI or address from which the request was retargeted,

5.3) The reason for the Request-URI or address modification,

5.4) Chronological ordering of the Request History information.

6) REQUEST-VALIDITY-req: Request-History is applicable to requests not sent within an established dialog. (i.e. INVITE, REGISTER, MESSAGE, and OPTIONS).

7) BACKWARDS-req: Request-History information may be passed from the generating entity backwards towards the UAC. This is needed to enable services that inform the calling party about the dialog establishment attempts.

8) FORWARDS-req: Request-History information may also be included by the generating entity in the request, if it is forwarded onwards.

2.1 Security Requirements

The Request History information is being inserted by a network element retargeting a Request, resulting in a slightly different problem than the basic SIP header problem, thus requiring specific consideration. It is recognized that these security requirements can be generalized to a basic requirement of being able to secure information that is inserted by proxies.

The potential security problems include the following:
1) A rogue application could insert a bogus Request History entry either by adding an additional entry as a result of retargeting or entering invalid information.

2) A rogue application could re-arrange the Request History information to change the nature of the end application or to mislead the receiver of the information.

Thus, a security solution for "Request History" must meet the following requirements:

1) SEC-req-1: The entity receiving the Request History must be able to determine whether any of the previously added Request History content has been altered.

2) SEC-req-2: The ordering of the Request History information must be preserved at each instance of retargeting.

3) SEC-req-3: The entity receiving the information conveyed by the Request History must be able to authenticate the source of the information.

4) SEC-req-4: To ensure the confidentiality of the Request History information, only entities which process the request should have visibility to the information.

It should be noted that these security requirements apply to any entity making use of the Request History information, either by retargeting and capturing the information, or as an application making use of the information received in either a Request or Response.

2.2 Privacy Requirements

Since the Request URI that is captured could inadvertently reveal information about the originator, there are general privacy requirements that MUST be met:

1) PRIV-req-1: The entity retargeting the Request must ensure that it maintains the network-provided privacy (as described in [RFC3323]) associated with the Request as it is retargeted.

2) PRIV-req-2: The entity receiving the Request History must maintain the privacy associated with the information.

In addition, local policy at a proxy may identify privacy requirements associated with the Request URI being captured in the Request History information.

3) PRIV-req-3: Request History information subject to privacy requirements shall not be included in outgoing messages unless it is protected as described in [RFC3323].

3. Request History Information Description

The fundamental functionality provided by the request history information is the ability to inform proxies and UAs involved in processing a request about the history or progress of that request [CAPABILITY-req]. The solution is to capture the Request-URIs as a request is forwarded in a new header for SIP messages: History-Info [CONTENT-req]. This allows for the capturing of the history of a request that would be lost with the normal SIP processing involved in the subsequent forwarding of the request. This solution proposes no changes in the fundamental determination of request targets or in the request forwarding as defined in sections 16.5 and 16.6 of the SIP protocol specification [RFC3261].

The History-Info header can appear in any request not associated with an established dialog, which includes INVITE, REGISTER, MESSAGE, REFER and OPTIONS [REQUEST-VALIDITY-req] and any valid response to these requests.[ISSUER-req]

The History-Info header is added to a Request when a new request is created by a UAC or Proxy, or when the target of a request is changed. The term 'retarget' is introduced to refer to this changing of the target of a request and the subsequent forwarding of that request. It should be noted that retargeting only occurs when the Request-URI indicates a domain for which the processing entity is responsible. In terms of the SIP protocol, the processing associated with retargeting is described in sections 16.5, and 16.6 of [RFC3261]. As described in section 16.5 of [RFC3261], it is possible for the target of a request to be changed by the same proxy multiple times (referred to as 'internal retargeting' in section 2), as the proxy MAY add targets to the target set after beginning Request Forwarding. Section 16.6 of [RFC3261] describes Request Forwarding. It is during this process of Request Forwarding, that the History Information is captured as an optional, additional header field. Thus, the addition of the History-Info header does not impact fundamental SIP Request Forwarding. An entity (UA or proxy) changing the target of a request in response to a redirect or REFER SHOULD also propagate any History-Info header from the initial Request in the new request [GENERATION-req, FORWARDS-req].

3.1 Optionality of History-Info

The History-Info header is optional in that neither UAs nor Proxies are required to support it. A new Supported header, Histinfo, is included in the Request to indicate whether the History-Info header is returned in Responses [BACKWARDS-req]. In addition to the Histinfo Supported header, local policy determines whether or not the header is added to any request, or for a specific Request-URI, being retargeted. It is possible that this could restrict the applicability of services which make use of the Request History Information to be limited to retargeting within domain(s) controlled by the same local policy, or between domain(s) which negotiate policies with other domains to ensure support of the given policy, or services for which "complete" History Information isn't required to provide the service. [OPTIONALITY-req] All applications making use of the History-info header MUST clearly define the impact of the information not being available and specify the processing of such a request.

3.2 Securing History-Info

This draft defines a new header for SIP. The draft RECOMMENDs the use of TLS as a mandatory mechanism to ensure the overall confidentiality of the History-Info headers [SEC-req-4]. This results in History-Info having at least the same level of security as other headers in SIP which are inserted by intermediaries. With the level of security provided by TLS [SEC-req-3], the information in the History-Info header can thus be evaluated to determine if information has been removed by evaluating the indices for gaps [SEC-req-1, SEC-req-2].

It would be up to the application to define whether it can make use of the information in the case of missing entries.

A more robust security solution would need to consider the aspects of the problem that are different than the hop by hop security problem solved by TLS, as each hop is not required to add the History-Info header. History-Info also introduces a slightly different problem than the basic SIP header or Identity [SIPATHID] problems, which is focused on securing the information in the initial request end to end. The History-Info header is being inserted by an entity as it targets and forwards a Request, thus the requirements for the security solution are similar to the Via and Record-Route headers. For the History-Info header, the general requirement is to secure a header that is inserted by an intermediary and then subsequently referenced, by other intermediaries to build the next header entry, or by an end application using the information to provide a service.

Thus, the general requirement for a more robust security solution for SIP takes the form of a middle to middle and middle to end security solution, which is addressed in a separate document [SIPIISEC]. The use of the middle-to-end security solution discussed in [SIPIISEC] allows the integrity of the History-Info to be ascertained as it traverses the intermediaries. Thus, including the History-Info header in SIP Requests and securing in this manner would add an additional level of security end to end, assuring the initiator of a Request that it has indeed reached the intended recipient.

3.3 Ensuring the Privacy of History-Info

Since the History-Info header can inadvertently reveal information about the requestor as described in [RFC3323], the Privacy header SHOULD be used to determine whether an intermediary can include the History-Info header in a Request that it receives and forwards [PRIV-req-2] or that it retargets [PRIV-req-1]. Thus, the History-Info header SHOULD not be included in Requests where the requestor has indicated a priv-value of Session or Header level privacy.

In addition, the History-Info header can reveal general routing information, which may be viewed by a specific intermediary or network, to be subject to privacy restrictions. Thus, local policy MAY also be used to determine whether to include the History-Info header at all, whether to capture a specific Request-URI in the header, or whether it be included only in the Request as it is retargeted within a specific domain. [PRIV-req-3] This is accomplished by adding a new priv-value to the Privacy header [RFC 3323] indicating whether any or a specific History-Info header(s) SHOULD be forwarded.

It is recognized that satisfying the privacy requirements can impact the functionality of this solution by overriding the request to generate the information. As with the optionality and security requirements, applications making use of History-Info SHOULD address any impact this may have.

4 Request History Information Protocol Details

This section contains the details and usage of the proposed new SIP protocol elements. It also discusses the security aspects of the solution.

4.1 Protocol Structure of History-Info

History-Info is a header field as defined by [RFC3261]. It is an optional header field and MAY appear in any request or response not associated with a dialog or which starts a dialog. For example, History-Info MAY appear in INVITE, REGISTER, MESSAGE, REFER and OPTIONS and any valid responses, plus NOTIFY requests which initiate a dialog.

The History-Info header carries the following information, with the mandatory parameters REQUIRED when the header is included a request or response:

- Targeted-to-URI (hi-targeted-to-uri): A mandatory parameter for capturing the Request URI for the specific Request as it is forwarded.

- Index (hi-index): A mandatory parameter for History-Info reflecting the chronological order of the information, indexed to also reflect the forking and nesting of requests. The format for this parameter is a string of digits, separated by dots to indicate the number of forward hops and retargets. This results in a tree representation of the history of the request, with the lowest level index reflecting a branch of the tree. By including the index and securing the header, the ordering of the History-info headers in the request is assured.[SEC-req-2] In addition, applications MAY extract a variety of metrics (total number of retargets, total number of retargets from a specific branch, etc.) based upon the index values.

- Reason: An optional parameter for History-info, reflected in the History-Info header by including the Reason Header [RFC3326] escaped in the Request URI being retargeted. A reason is not included for a Request URI when it is first added in a History-info header, but rather is added when that particular Request-URI is retargeted. Note, that this does appear to complicate the security problem, however, retargeting only occurs when the

Request-URI indicates a domain for which the processing entity is responsible, thus it would be the same processing entity that initially added the Request-URI to the header that would be updating it with the Reason.

- o Privacy: An optional parameter for History-info, reflected in the History-Info header by including the Privacy Header [RFC3323] with a priv-value of "history" escaped in the Request URI or by adding the Privacy header with a priv-value of "history" to the Request. The use of the Privacy Header with a priv-value of "history" indicates whether a specific or all History-Info headers SHOULD NOT be forwarded.

- o Extension (hi-extension): An optional parameter to allow for future optional extensions. As per the [RFC3261], any implementation not understanding an extension SHOULD ignore it.

The following summarizes the syntax of the History-Info header, based upon the standard SIP syntax [RFC3261]:

```
History-Info = "History-Info" HCOLON

                    hist-info *(COMMA hist-info)

hist-info = hi-targeted-to-uri *( SEMI hi-param )

hi-targeted-to-uri= name-addr

hi-param = hi-index / hi-extension

 hi-index = "index" EQUAL 1*DIGIT *(DOT 1*DIGIT)

hi-extension = generic-param
```

This document adds the following entry to Table 2 of [RFC3261]. Additions to this table are also provided for extension methods at the time of publication of this document. This is provided as a courtesy to the reader and is not normative in any way.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	MSG
History-Info		amdr	-	-	-	o	o	o	o

Header field	where	proxy	SUB	NOT	REF	INF	UPD	PRA	PUB
History-Info		amdr	-	o	o	-	-	-	-

4.2 Protocol Examples

The following provides some examples of the History-Info header. Note that the backslash, CRLF, and spacing between the fields in the examples below are for readability purposes only.

```
History-Info:<sip:UserA@ims.example.com?Reason=SIP;\
  cause=302;text="Moved Temporarily">; index=1; foo=bar

History-Info: <sip:UserA@ims.example.com?Reason=SIP;\
   cause=302; text="Moved Temporarily">; index=1.1,
   <sip:UserB@example.com?Privacy=history&Reason=SIP;cause=486;\
   text="Busy Here">;index=1.2,
   <sip:45432@vm.example.com>;index=1.3
```

4.3 Protocol usage

This section describes the processing specific to UAs and Proxies for the History-Info header, the Histinfo option tag and the priv-value of "history". As discussed in section 1, the fundamental objective is to capture the target Request-URIs as a request is forwarded. This allows for the capturing of the history of a request that would be lost due to subsequent (re)targeting and forwarding. To accomplish this for the entire history of a request, either the UAC must capture the Request-URI in the initial request or a proxy must add History-Info headers for both the Request-URI in the initial request and the target Request-URI as the request is forwarded. The basic processing is for each entity forwarding a request to add a History-Info header for the target Request-URI, updating the index and adding the Reason as appropriate for any retargeted Request-URI.

4.3.1 UAC Behavior

The UAC SHOULD include the Histinfo option tag in the Supported header in any request not associated with an established dialog for which the UAC would like the History-Info in the Response. In addition, the UAC SHOULD initiate the capturing of the History Information by adding a History-Info header using the Request-URI of the request as the hi-targeted-to-uri and initializing the index to the RECOMMENDED value of 1 in the History-Info header.

In the case where the request is routed to a redirect server and the UAC receives a 3xx response with a Contact header, the UAC MAY maintain the previous History-Info entry(-ies) in the request. A new History-Info entry MAY then be added for the URI from the Contact header (which will become the new Request-URI). In this case, the

index is created by reading and incrementing the value of the index from the previous history entry, thus following the same rules as those prescribed for a proxy in retargeting, described in section 4.3.3.1.3. An example of this scenario can be found in Appendix D.

A UAC that does not want History-Info headers added due to privacy considerations SHOULD include a Privacy header with a priv-value(s) of "session", "header" or "history" in the request.

The processing of the History-Info header received in the Response is application specific and outside the scope of this draft. However, the validity of the information SHOULD be ensured prior to any application usage. For example, the entries MAY be evaluated to determine gaps in indices, which could indicate that an entry has been maliciously removed or removed for privacy reasons. Either way, an application MAY want to be notified of potentially missing information.

4.3.2 UAS Behavior

The processing of the History-Info header by a UAS in a Request depends upon local policy and specific applications at the UAS which might make use of the information. Prior to any application usage of the information, the validity SHOULD be ascertained. For example, the entries MAY be evaluated to determine gaps in indices, which could indicate that an entry has been maliciously removed or removed for privacy reasons. Either way, an application MAY want to be notified of potentially missing information.

If the Histinfo option tag is received in a request, the UAS should include any History-Info received in the request in the subsequent response.

4.3.3 Proxy Behavior

The inclusion of the History-Info header in a Request does not alter the fundamental processing of proxies for determining request targets as defined in section 16.5 of [RFC3261]. Whether a proxy adds the the History-Info header as it forwards a Request depends upon the following considerations:

1. Whether the Request contains the Histinfo option tag in the Supported header.
2. Whether the proxy supports the History-Info header.
3. Whether the Request contains a Privacy header with a priv-value of "session", "header" or "history".
4. Whether any History-Info header added for a proxy/domain should go outside that domain. An example being the use of

the History-Info header within the specific domain in which it is retargeted, however, policies (for privacy, user and network security, etc.) prohibit the exposure of that information outside that domain. A proxy MAY insert the Privacy header with a priv-value of "history" to indicate this. An example of such an application is provided in Appendix C.

5. Whether the History-Info header is added for a specific Request URI due to local privacy policy considerations. A proxy MAY add the Privacy header with a priv-value of "history" associated with the specific hi-targeted-to-uri.

An example policy would be a proxy that only adds the History-Info header if the Histinfo option tag is in the Supported header. Other proxies may have a policy that they always add the header, but never forward it outside a particular domain, accomplishing this by adding a Privacy header with a priv-value of "history" to allow the information to be collected for internal retargeting only.

Each application making use of the History-Info header SHOULD address the impacts of the local policies on the specific application (e.g. what specification of local policy is optimally required for a specific application and any potential limitations imposed by local policy decisions).

Consistent with basic SIP processing of optional headers, proxies SHOULD maintain History-Info headers, received in messages being forwarded, independent of whether local policy supports History-Info.

The specific processing by proxies for adding the History-Info headers in Requests and Responses is described in detail in the following sections.

4.3.3.1 Adding the History-Info header to Requests

Upon evaluation of the considerations under which the History-Info header is to be included in requests (e.g. no Privacy header overriding inclusion, local policy supports, etc.), detailed in section 4.3.3, a proxy SHOULD add a History-Info header as it forwards a Request. Section 16.6 of [4] defines the steps to be followed as the proxy forwards a Request. Step 5 prescribes the addition of optional headers. Although, this would seem the appropriate step for adding the History-info header, the interaction with Step 6 "Postprocess routing information" and the impact of a strict route in the Route header could result in the Request-URI being changed, thus adding the History-info header between steps 8 (adding Via header) and 9 (adding Content-Length) is RECOMMENDED. Note, that in the case of loose routing, the Request-URI does not change during the forwarding of a Request, thus the capturing of

History-Info for such a request would result in duplicate Request-URIs with different indices. The History-Info header SHOULD be added following any History-Info header received in the request being forwarded. Additionally, if a request is received that doesn't include a History-Info header, the proxy MAY add an additional History-Info header preceding the one being added for the current request being forwarded. The index for this entry is RECOMMENDED to start at 1. The following subsections define the details of creating the information associated with and in the History-Info header.

4.3.3.1.1 Privacy in the History-Info header

If the proxy's local policies, per consideration 4 in section 4.3.3, indicate that this History-Info entry and any entries added due to subsequent retargeting should not be forwarded beyond the domain for which this intermediary is responsible, then a Privacy header with a priv-value of "history" SHOULD be added to the request, if there is not already one, provided the request is being forwarded to a specific URI associated with the domain(s) for which this entity is responsible.

If a request is being forwarded to a Request URI associated with a domain for which the proxy is not responsible, the proxy needs to determine if there are any entries to be removed prior to forwarding. Any headers associated with the domain(s) for which this proxy is responsible SHOULD be removed prior to forwarding.

If through local policy, there is knowledge of privacy associated with a specific URI being captured as the hi-targeted-to-uri, a Privacy header with a priv-value of "history" SHOULD be associated with this specific URI as the request is forwarded, if it is being forwarded to a Request URI associated with a domain for which the processing entity is responsible.

If a request is being forwarded to a Request URI, for which the processing entity is not responsible, the proxy needs to determine if there are any entries, that need to be removed prior to forwarding. The proxy needs to determine if any of the specific URIs that have been captured in the History-Info entries, associated with the domain(s) for which it is responsible, have a priv-value of "history". Each of these header entries SHOULD be removed from the Request prior to forwarding.

4.3.3.1.2 Reason in the History-Info header

For retargets that are the result of an explicit SIP response, the SIP Response Code that triggered the retargeting MUST be included in the Reason header field of the Request URI that has been retargeted. This should occur prior to the forwarding of the request, as it

associated with the previous hi-targeted-to-uri, since it reflects the reason why the Request to that specific URI was not successful.

For retargets as a result of timeouts or internal events, a Reason MAY be included in the Reason header field of the Request URI that has been retargeted.

4.3.3.1.3 Indexing in the History-Info header

In order to maintain ordering and accurately reflect the nesting and retargeting of the request, an index MUST be included along with the Targeted-to-URI being captured. Per the ABNF in section 4.1, the index consists of a dot delimited series of digits (e.g. 1.1.2), with each dot reflecting the number of hops or level of nesting of the request. Within each level, the number reflects the number of peer entities to which the request has been routed. Thus, the indexing results in a logical tree representation for the history of the Request. It is recommended that for each level of indexing, the index start at 1. It is recommended that an increment of 1 is used for advancing to a new branch. For retargets within a proxy, the proxy MUST maintain the current level of nesting by incrementing by 1 the lowest/last digit of the index for each instance of retargeting, thus reflecting the number of retargets (branches) within the proxy.

The basic rules for adding the index are summarized as follows:

1. Basic Forwarding: In the case of a Request that is being forwarded, the index is determined by adding another level of indexing since the depth/length of the branch is increasing. To accomplish this, the proxy reads the value from the History-Info header in the received request, if available, and adds another level of indexing by appending the DOT delimiter followed by an initial index for the new level RECOMMENDED to be 1. For example, if the index in the last History-Info header field in the received request is 1.1, this proxy would initialize its index to 1.1.1 and forward the request.

2. Retargeting within a Proxy - 1st instance: For the first instance of retargeting within a Proxy, the calculation of the index follows that prescribed for basic forwarding.

3. Retargeting within a Proxy - subsequent instance: For each subsequent retargeting of a request by the same proxy, another branch is added. With the index for each new branch calculated by incrementing the last/lowest digit at the current level, thus the index in the next request forwarded by this same proxy, following the example above, would be 1.1.2.

4. Retargeting based upon a Response: In the case of retargeting due to a specific response (e.g. 302), the index would be calculated per rule 3. That is, the lowest/last digit of the index is incremented (i.e. a new branch is created), with the increment RECOMMENDED to be 1. For example, if the index in the History-Info header of the received request was 1.2, then the index in the History-Info header field for the new hi-targeted-to-URI would be 1.3.

5. Retargeting the request in parallel: If the request forwarding is done in parallel, the index MUST be captured for each forked request per the rules above, with each new Request having a unique index. The only difference in the messaging for this scenario and the messaging produced per basic proxy retargeting in rules 2 and 3 is these forwarded requests do not have History-Info entries associated with their peers. The proxy builds the subsequent response (or request) using the amalgamated information associated with each of those requests and including the header entries in the order indicated by the indexing. Section 4.5 provides an example of a parallel request scenario, highlighting this indexing mechanism.

4.3.3.2 Processing History-Info in Responses

A proxy that receives a Request with the Histinfo option tag in the Supported header, and depending upon a local policy supporting the capture of History-Info, SHOULD return captured History-Info in subsequent, provisional and final responses to the Request.

It should be noted that local policy considerations, for network and intermediary privacy, MAY restrict the sending of the History-Info headers added by the intermediary in subsequent responses. Thus, in such cases, the proxy MAY remove from these responses the History-Info headers which it inserted in the original forwarded request.

4.3.4 Redirect Server Behavior

A redirect server SHOULD NOT add any new History-Info, as that would be done by the entity receiving the 3xx response. However, a redirect server MAY include History-Info in responses by adding any History-Info headers received in a request to a subsequent response.

4.4 Security for History-Info

As discussed in Section 1, the security requirements are partially met by recommending the use of TLS (a basic SIP requirement per [RFC3261]) for hop by hop security. In addition, the use of the middle-to-end security solution discussed in [SIPIISEC] allows the

integrity of the History-Info to be ascertained as it traverses the intermediaries.

4.5 Example Applications using History-Info

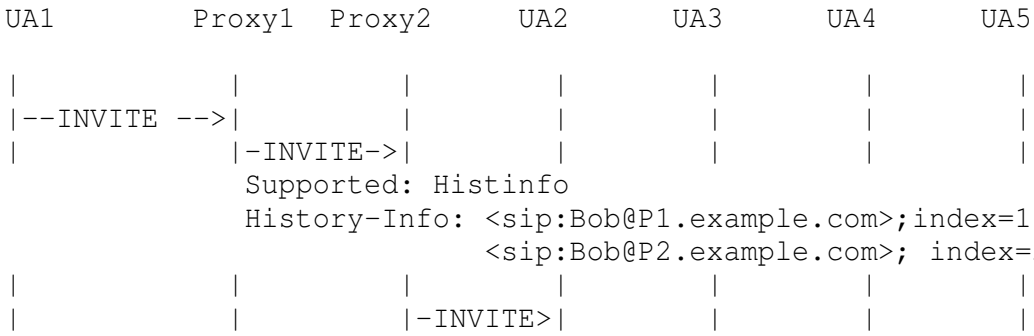
This scenario highlights an example where the History-Info in the response is primarily of use in not retrying routes that have already been tried by another proxy. Note, that this is just an example and that there may be valid reasons why a Proxy would want to retry the routes and thus, this would likely be a local proxy or even user specific policy.

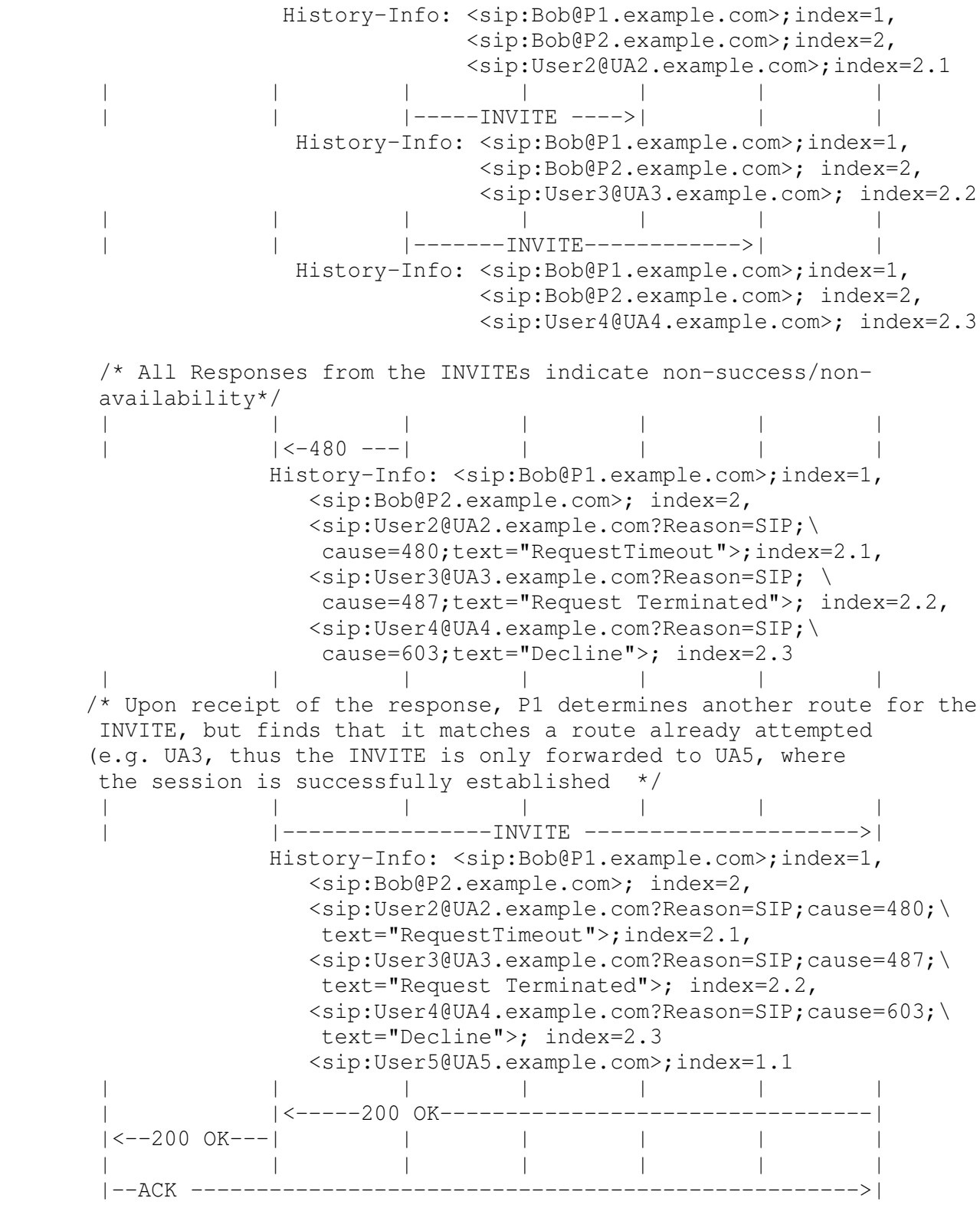
UA 1 sends a call to "Bob" to proxy 1. Proxy 1 forwards the request to Proxy 2. Proxy 2 sends the requests in parallel and tries several places (UA2, UA3 and UA4) before sending a response to Proxy 1 that all the places are busy. Proxy 1, without the History-Info, would try several some of the same places (e.g. UA3) based upon registered contacts for "Bob", before completing at UA5. However, with the History-Info, Proxy 1 determines that UA3 has already received the invite, thus the INVITE goes directly to UA5.

Section 4.5.1 provides this same scenario using one of the privacy mechanism, with Proxy2 adding the Privacy header indicating that the History-Info header is not to be propagated outside P2's domain. This scenario highlights the potential functionality lost with the use of "history" privacy in the Privacy header for the entire request and the need for careful consideration on the use of privacy for History-Info.

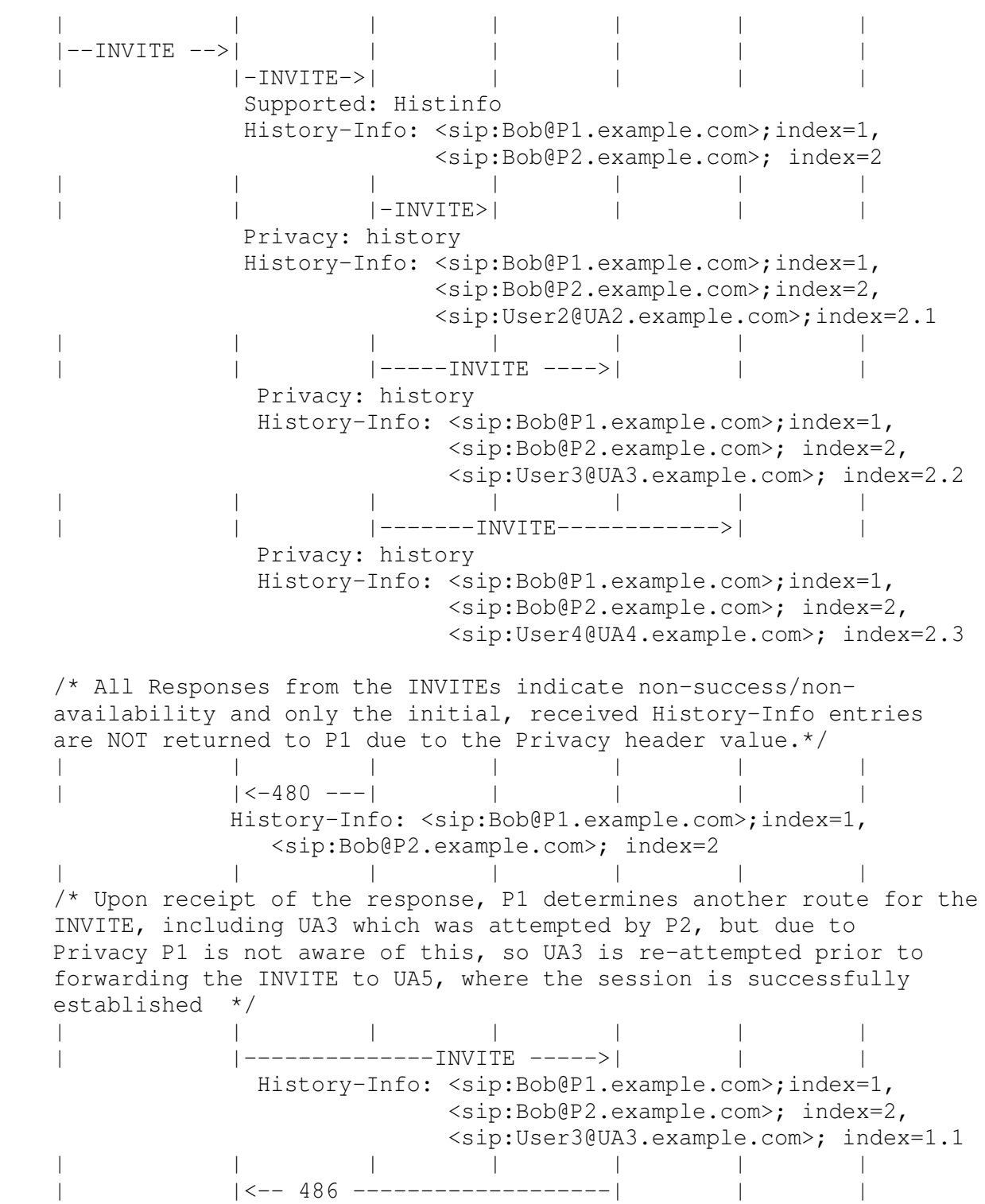
Section 4.5.2 also provides the same scenario using one of the privacy mechanisms, however, due to local policy at Proxy2, only one of the Request-URIs (UA4) in the History-Info contains a priv-value of "history", thus allowing some optimized functionality in the routing of the request, but still maintaining privacy for specific URIs.

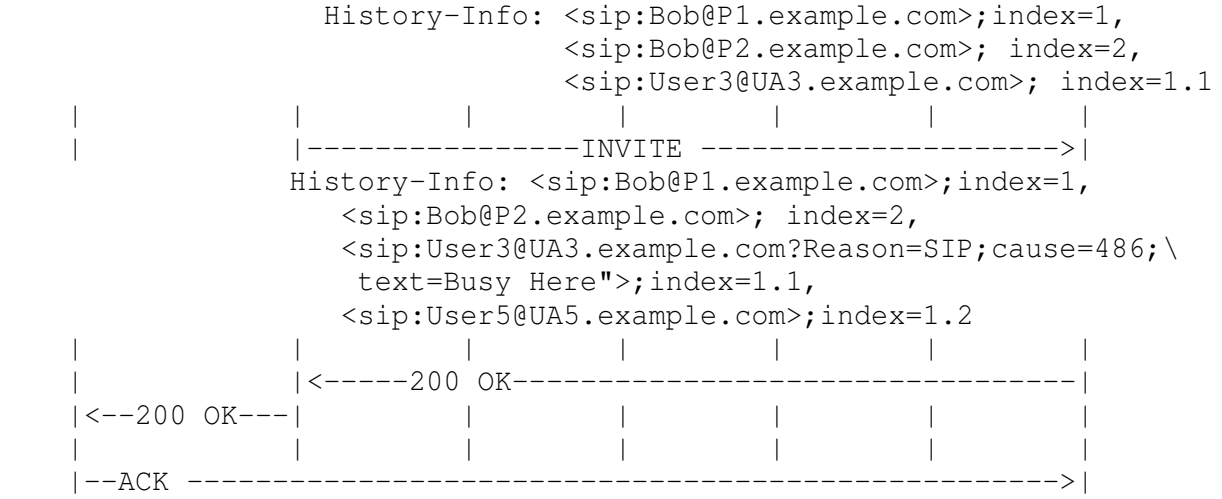
Additional detailed scenarios are available in the appendix.



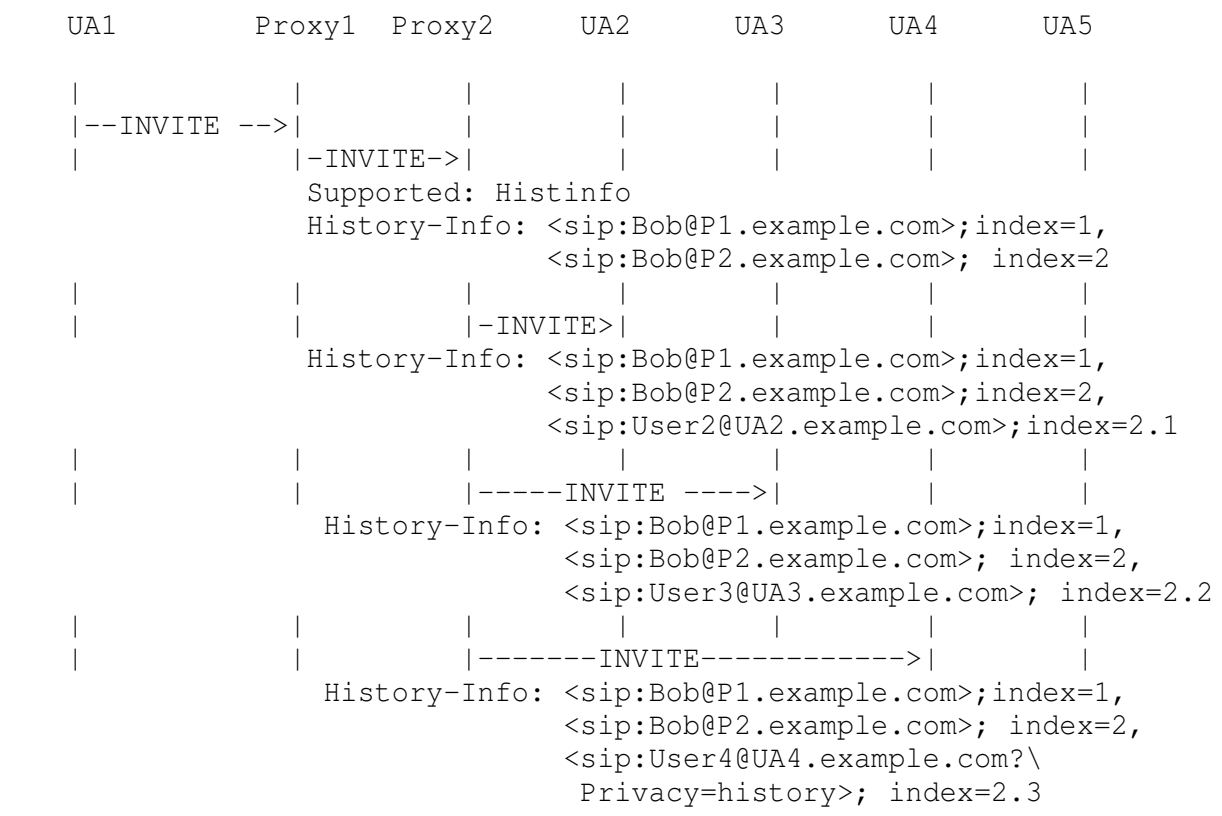

History-Info: <sip:Bob@P1.example.com>;index=1,
<sip:Bob@P2.example.com>;index=2,
<sip:User2@UA2.example.com>;index=2.1
-----INVITE ---->
History-Info: <sip:Bob@P1.example.com>;index=1,
<sip:Bob@P2.example.com>; index=2,
<sip:User3@UA3.example.com>; index=2.2
-------INVITE------------>
History-Info: <sip:Bob@P1.example.com>;index=1,
<sip:Bob@P2.example.com>; index=2,
<sip:User4@UA4.example.com>; index=2.3
/* All Responses from the INVITEs indicate non-success/non-
availability*/
<-480 ---
History-Info: <sip:Bob@P1.example.com>;index=1,
<sip:Bob@P2.example.com>; index=2,
<sip:User2@UA2.example.com?Reason=SIP;\
cause=480;text="RequestTimeout">;index=2.1,
<sip:User3@UA3.example.com?Reason=SIP; \
cause=487;text="Request Terminated">; index=2.2,
<sip:User4@UA4.example.com?Reason=SIP;\
cause=603;text="Decline">; index=2.3
/* Upon receipt of the response, P1 determines another route for the
INVITE, but finds that it matches a route already attempted
(e.g. UA3, thus the INVITE is only forwarded to UA5, where
the session is successfully established */
----------------INVITE ---------------------->
History-Info: <sip:Bob@P1.example.com>;index=1,
<sip:Bob@P2.example.com>; index=2,
<sip:User2@UA2.example.com?Reason=SIP;cause=480;\
text="RequestTimeout">;index=2.1,
<sip:User3@UA3.example.com?Reason=SIP;cause=487;\
text="Request Terminated">; index=2.2,
<sip:User4@UA4.example.com?Reason=SIP;cause=603;\
text="Decline">; index=2.3
<sip:User5@UA5.example.com>;index=1.1
<-----200 OK----------
<--200 OK---
--ACK ---------->
Barnes Expires January 8, 2005 [Page 19]

4.5.1 Example with Privacy header for entire request at Proxy2





4.5.2 Example with Privacy header for specific URI (UA4) at Proxy2



```
                History-Info: <sip:Bob@P1.example.com>;index=1,
                   <sip:Bob@P2.example.com>; index=2,
                   <sip:User2@UA2.example.com?Reason=SIP;\
                    cause=480;text="RequestTimeout">;index=2.1,
                   <sip:User3@UA3.example.com?Reason=SIP; \
                    cause=487;text="Request Terminated">; index=2.2,
 |              |         |        |        |        |        |
/* Upon receipt of the response, P1 determines another route for the
 INVITE, but finds that it matches a route already attempted
(e.g. UA3), thus the INVITE is only forwarded to UA5, where
 the session is successfully established  */
 |              |         |        |        |        |        |
 |              |-----------------INVITE ---------------------->|
                History-Info: <sip:Bob@P1.example.com>;index=1,
                   <sip:Bob@P2.example.com>; index=2,
                   <sip:User2@UA2.example.com?Reason=SIP;cause=480;\
                    text="RequestTimeout">;index=2.1,
                   <sip:User3@UA3.example.com?Reason=SIP;cause=487;\
                    text="Request Terminated">; index=2.2,
                   <sip:User5@UA5.example.com>;index=1.1
 |              |         |        |        |        |        |
 |              |<-----200 OK--------------------------------|
 |<--200 OK---|         |        |        |        |        |
 |              |         |        |        |        |        |
 |--ACK ------------------------------------------------------>|
```

5. Application Considerations

As seen by the example scenarios in the appendix, History-Info provides a very flexible building block that can be used by intermediaries and UAs for a variety of services. As such, any services making use of History-Info must be designed with the following considerations:

1) History-Info is optional, thus a service should define default behavior for requests and responses not containing History-Info headers.
2) History-Info may be impacted by privacy considerations. Applications requiring History-Info need to be aware that if Header, Session or History level privacy is requested by a UA (or imposed by an intermediary) that History-Info may not be available in a request or response. This would be addressed by an application in the same manner as the previous consideration by ensuring there is reasonable default behavior should the information not be available.
3) History-Info may be impacted by local policy. Each application making use of the History-Info header SHOULD address the impacts of the local policies on the specific application (e.g. what specification of local policy is optimally required for a

specific application and any potential limitations imposed by local policy decisions). Note, that this is related to the optionality and privacy considerations identified in 1 and 2 above, but goes beyond that. For example, due to the optionality and privacy considerations, an entity may receive only partial History-Info entries; will this suffice? Note, that this would be a limitation for debugging purposes, but might be perfectly satisfactory for some models whereby only the information from a specific intermediary is required.

4) The security associated with the Request History Information is optional. Whether there is security applied to the entries depends upon local policy. The impact of lack of having the information compromised depends upon the nature of the specific application (e.g. is the information something that appears on a display or is it processed by automata which could have negative impacts on the subsequent processing of a request?). It is suggested that the impact of an intermediary not supporting the security recommendations should be evaluated by the application to ensure that the impacts have been sufficiently addressed by the application.

6. Security Considerations

This draft provides a proposal in sections 3.2 and 4.4 for addressing the Security requirements identified in section 2.1 by mandating the use of TLS between entities. With TLS, History-Info headers are no less, nor no more, secure than other SIP headers, which generally have even more impact on the subsequent processing of SIP sessions than the History-Info header. A more robust security solution, which would secure headers added by proxies, SHOULD be used for History-Info implementations once there is a solution to the requirements identified in [SIPIISEC].

7. IANA Considerations

(Note to RFC Editor: Please fill in all occurrences of XXXX in this section with the RFC number of this specification).

7.1 Registration of new SIP History-Info header

This document defines a new SIP header field name: History-Info and a new option tag: Histinfo.

The following changes should be made to http:///www.iana.org/assignments/sip-parameters

The following row should be added to the header field section:

Header Name	Compact Form	Reference

History-Info	none	[RFCXXXX]

The following should be added to the Options Tags section:

Name	Description	Reference
Histinfo	When used with the Supported header, this option tag indicates support for the History Information to be captured for requests and returned in subsequent responses. This tag is not used in a Proxy-Require or Require header field since support of History-Info is optional.	[RFCXXXX]

7.2 Registration of "history" for SIP Privacy header

This document defines a new priv-value for the SIP Privacy header: history

The following changes should be made to http://www.iana.org/assignments/sip-priv-values

The following should be added to the registration for the SIP Privacy header:

Name	Description	Registrant	Reference
history	Privacy requested for History-Info header(s)	Mary Barnes mary.barnes@nortelnetworks.com	[RFCXXXX]

Changes since last version

Changes from the û02 to the û03 version:

- Editorial changes: Updating to the new template to reflect new IPR guidelines, ensuring that the normative text is complete and accurate in section 4.1, removing "Editor's Notes", etc.
- Section 4.5: Fixed error in cause (408 -> 480).
- Examples: changed the domain to "example.com", IP addresses to the 192.0.2.0/24 range, changed occurrences of "Reason:" to "Reason=", added use of Privacy header to examples.
- Added text to reflect WG consensus on Issue-1: Privacy indication for History-Info entries. Proposed an extension to the priv-values defined in RFC 3323 in abstract and section 3.3, impacting the protocol structure in section 4.1 and

processing in 4.3.3 (and 4.3.3.1 and 4.3.3.2). In addition, the new priv-value needs to be registered with IANA, per section 7.
- o Removed Open Issues section. For Issue-2, there was not WG consensus to define an algorithm for bounding the number of History-Info entries, but rather that is left as an implementation decision.
- o Updated Security discussions to reflect WG consensus that TLS is mandatory and sufficient for general History-Info implementation. The e2m and m2m security solutions can be applied to History-Info when they become available to provide a more robust SIP solution.
- o Section 4.1: Added additional text to ensure that all the information in the History-Info header is appropriately and normatively described (in text).
- o Added text in section 4.3.1 and an example to the appendices to address the UAC having added multiple History-Info headers for the case where the 3xx response goes back to the UAC and it's the UAC that retargets the INVITE request.
- o Clarified the addition of the Reason header in section 4.3.3.1.2.
- o Further delineated the basic rules in section 4.3.3.1.3 for calculating the index for various scenarios, as this was still causing some confusion.

Changes from the û01 to the û02 version:

- o Merged the SIPPING WG requirements draft into this document. Note that this increments the section references in the remainder of the document by 2 (and by 3 for Security and IANA considerations due to new section added). Also, removed redirect server from ISSUER-req since the solution identified this as not being required (or desirable).
- o Added an explicit privacy requirement (PRIV-req-3) for the proxy's role in recognizing and maintaining privacy associated with a Request-URI being captured in History-Info due to local policy. (Note, that the text was already there, it just wasn't highlighted as an explicit requirement).
- o Clarified the use of CRLF and spacing in the example headers in section 4.2.
- o Removed the compact form for the header since unknown headers with multiple entries would not be recognized (i.e. this may cause parsing problems).
- o Added a summary of Application Considerations to address concerns about the optional usage of History-Info.
- o Converted the references from numbers to labels to avoid the continual problem of renumbering.
- o Minor editorial changes (per NITS highlighted by Rohan and Eric and some minor rewording for clarity).

Changes from the û00 to the û01 version:

- o Attempted to be more explicit about the fundamental processing associated with the header. Removed definitions of new terms, only referencing the terms from the requirements in the context of the fundamental SIP processing implied by the terms.
- o Attempted to clarify the Index and the related processing.
- o Added more detail addressing the privacy requirements.
- o Added a bit more detail on security. The security solution remains in a separate document and this document will need updating once that is completed.
- o Updated the examples (in section 2.5 and appendix) and clarified the definition and the maintenance of the Index in sections 2.1 and 2.3.3.1.
- o Clarified the Reason description in section 2.1. There had been an error in the description of the processing that was a remnant of the change to include only a single URI for each History-Info header.
- o Miscellaneous editorial changes (i.e. HistInfo -> Histinfo, etc.)

Changes from individual draft-barnes-sipping-history-info-02 to the û 00 WG version:

- o Updated references and added reference to Security solution draft.
- o Removed appendix D which included background on analysis of solution options.
- o Cleaned up the document format per rfc2223bis.
- o Strengthened the inclusion of the INDEX as a MUST (per discussion at IETF-56).
- o Added text around the capturing of the Reason (SHOULD be captured for SIP responses and MAY be captured for other things such as timeouts).
- o Clarified the response processing 2.3.3.2 to include provisional responses and the sending of a 183 to convey History-Info.
- o Added section 2.3.4 to address Redirect Server behavior.

Normative References

[RFC3261] J. Rosenberg et al, "SIP: Session initiation protocol," RFC 3261, June, 2002.

[RFC3326] H. Schulzrinne, D. Oran, G. Camarillo, "The Reason Header Field for the Session Initiation Protocol", RFC 3326, December, 2002.

[RFC3323] J. Peterson, "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November, 2002.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[SIPIISEC] M. Barnes, "A Mechanism to Secure SIP Headers Inserted by Intermediaries", draft-barnes-sipping-inserted-info-01.txt, October, 2003.

Informational References

[SIPSVCEX] A. Johnson, "SIP Service Examples", draft-ietf-sipping-service-examples-05.txt, November, 2002.

[SIPATHID] J. Peterson, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-01.txt, February, 2003.

[RFC3665] A. Johnson et al, "SIP Basic Call Flow Examples", RFC 3665, BCP 75, December, 2003.

Acknowledgements

The editor would like to acknowledge the constructive feedback provided by Robert Sparks, Paul Kyzivat, Scott Orton, John Elwell, Nir Chen, Francois Audet, Palash Jain, Brian Stucker, Norma Ng, Anthony Brown, Jayshree Bharatia, Jonathan Rosenberg, Eric Burger and Martin Dolly.

The editor would like to acknowledge the significant input from Rohan Mahy on some of the normative aspects of the ABNF, particularly around the need for and format of the index and around the enhanced SIP security aspects enabled by this draft.

Contributors' Addresses

Cullen, Mark and Jon contributed to the development of the initial requirements.

Cullen and Mark provided substantial input in the form of email discussion in the development of the initial version of the individual solution document.

Cullen Jennings
Cisco Systems
170 West Tasman Dr
MS: SJC-21/3

Tel: +1 408 527 9132
Email: fluffy@cisco.com

Jon Peterson
NeuStar, Inc.
1800 Sutter Street, Suite 570
Concord, CA 94520
USA

Phone: +1 925-363-8720
EMail: Jon.Peterson@NeuStar.biz

Mark Watson
Nortel Networks (UK)
Maidenhead Office Park (Bray House)
Westacott Way
Maidenhead,
Berkshire
England

Tel: +44 (0)1628-434456
Email: mwatson@nortelnetworks.com

Author's Address

Mary Barnes
Nortel Networks
2380 Performance Drive
Richardson, TX USA

Phone: 1-972-684-5432
Email: mary.barnes@nortelnetworks.com

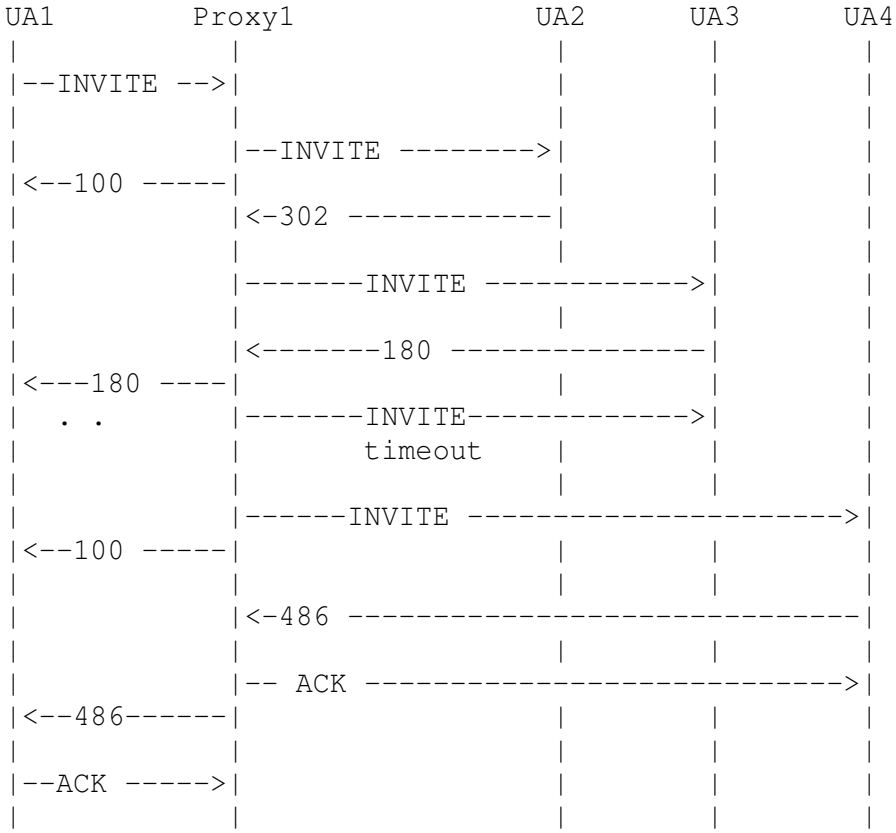
Appendix A Forking Scenarios

A.1 Sequentially forking (History-Info in Response)

This scenario highlights an example where the History-Info in the response is useful to an application or user that originated the request.

UA 1 sends a call to "Bob" via proxy 1. Proxy 1 sequentially tries several places (UA2, UA3 and UA4) unsuccessfully before sending a response to UA1.

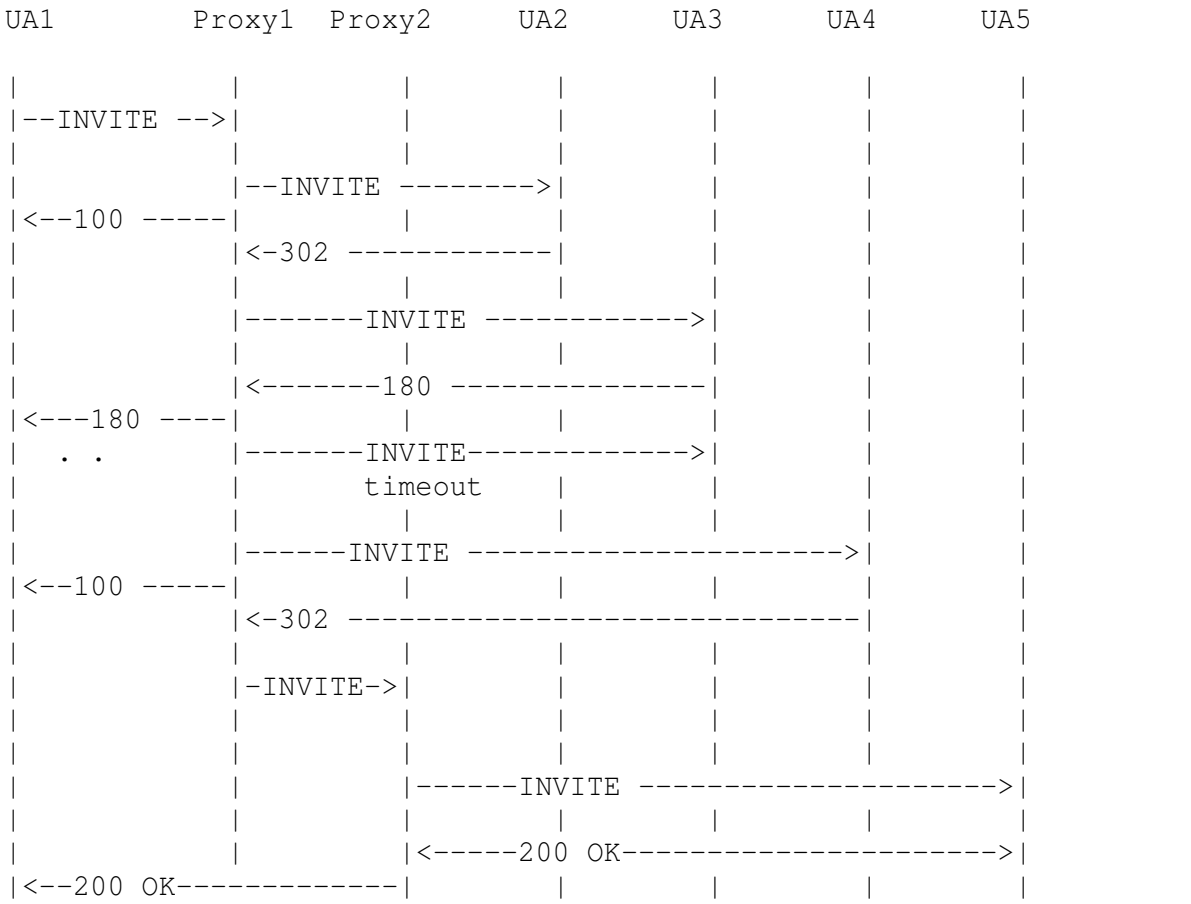
This scenario is provided to show that by providing the History-Info to UA1, the end user or an application at UA1 could make a decision on how best to attempt finding "Bob". Without this mechanism UA1 might well attempt UA3 (and thus UA4) and then re-attempt UA4 on a third manual attempt at reaching "Bob". With this mechanism, either the end user or application could know that "Bob" is busy on his home phone and is physically not in the office. If there were an alternative address for "Bob" known to this end user or application, that hasn't been attempted, then either the application or the end user could attempt that. The intent here is to highlight an example of the flexibility of this mechanism that enables applications well beyond SIP as it is certainly well beyond the scope of this draft to prescribe detailed applications.

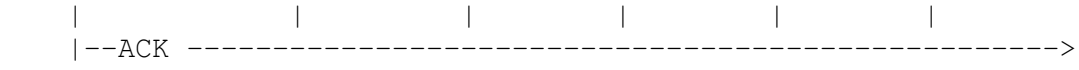


A.2 Sequential Forking (with Success)

This scenario highlights an example where the History-Info in the request is primarily of use in not retrying routes that have already been tried by another proxy. Note, that this is just an example and that there may be valid reasons why a Proxy would want to retry the routes and thus, this would like be a local proxy or even user specific policy.

UA 1 sends a call to "Bob" to proxy 1. Proxy 1 sequentially tries several places (UA2, UA3 and UA4) before retargeting the call to Proxy 2. Proxy 2, without the History-Info, would try several of the same places (UA3 and UA4)based upon registered contacts for "Bob", before completing at UA5. However, with the History-Info, Proxy 2 determines that UA3 and UA4 have already received the invite, thus the INVITE goes directly to UA5.



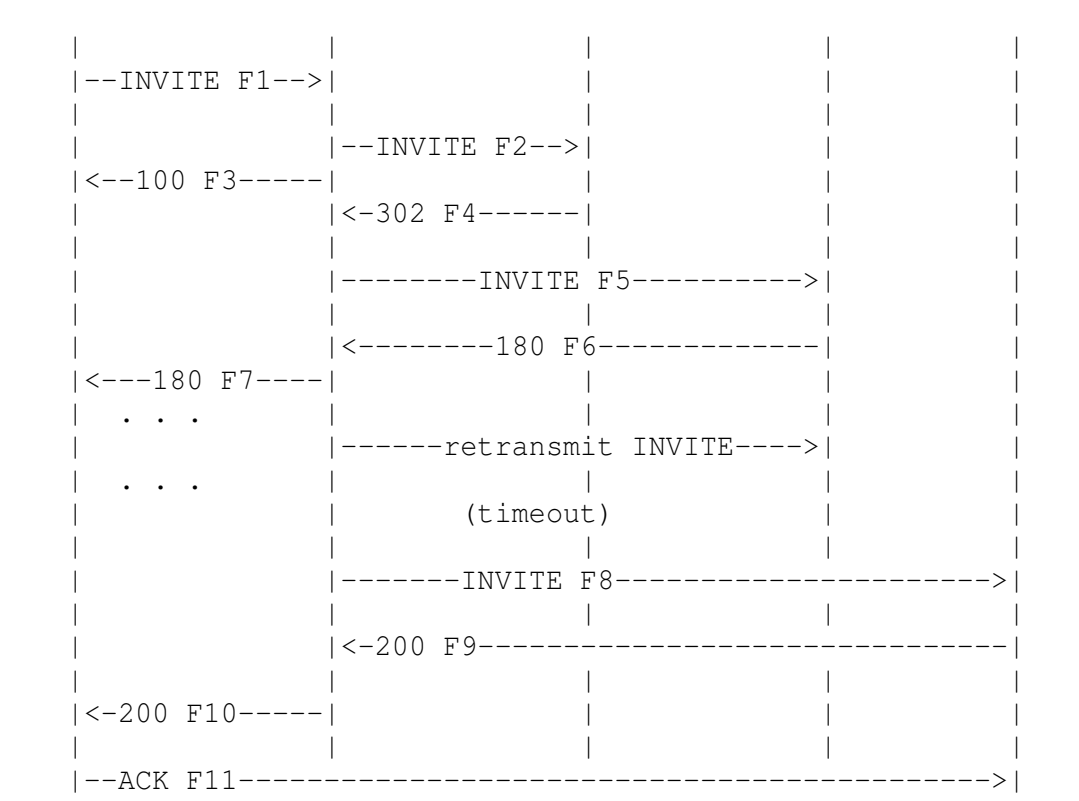


[Editor's Note: Need to add the details of the messages here.]

Appendix B Voicemail

This scenario highlights an example where the History-Info in the request is primarily of use by an edge service (e.g. Voicemail Server). It should be noted that this isn't intended to be a complete specification for this specific edge service as it is quite likely that additional information is need by the edge service. History-Info is just one building block that this service makes use of.

UA 1 called UA A which had been forwarded to UA B which forwarded to a UA VM (voicemail server). Based upon the retargeted URIs and Reasons (and other information) in the INVITE, the VM server makes a policy decision about what mailbox to use, which greeting to play etc.



```
INVITE F1   UA1->Proxy

INVITE sip:UserA@example.com SIP/2.0
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>
Call-Id: 12345600@here.com
CSeq: 1 INVITE
Contact: BigGuy <sip:User1@here.com>
Content-Type: application/sdp
Content-Length: <appropriate value>

 v=0
 o=UserA 2890844526 2890844526 IN IP4 client.here.com
 s=Session SDP
 c=IN IP4 192.0.2.3
 t=0 0
 m=audio 49170 RTP/AVP 0
 a=rtpmap:0 PCMU/8000

 /*Client for UA1 prepares to receive data on port 49170
 from the network. */

 INVITE F2 Proxy->UA-A

 INVITE sip:UserA@ims.example.com SIP/2.0
 Via: SIP/2.0/UDPims.example.com:5060;branch=1
   Via: SIP/2.0/UDP here.com:5060
 Record-Route: <sip:UserA@example.com>
 From: BigGuy <sip:User1@here.com>
 To: LittleGuy <sip:UserA@example.com>
 Call-Id: 12345600@here.com
 CSeq: 1 INVITE
 History-Info: <sip:UserA@ims.example.com>; index=1
 Contact: BigGuy <sip:User1@here.com>
 Content-Type: application/sdp
 Content-Length: <appropriate value>

 v=0
 o=UserA 2890844526 2890844526 IN IP4 client.here.com
 s=Session SDP
 c=IN IP4 192.0.2.3
 t=0 0
 m=audio 49170 RTP/AVP 0
 a=rtpmap:0 PCMU/8000

 100 Trying F3 Proxy->UA1

 SIP/2.0 100 Trying
```

```
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>
Call-Id: 12345600@here.com
CSeq: 1 INVITE
Content-Length: 0


302 Moved Temporarily F4  UserA->Proxy
SIP/2.0 302 Moved Temporarily
Via: SIP/2.0/UDP ims.example.com:5060;branch=1
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>;tag=3
Call-Id: 12345600@here.com
CSeq: 1 INVITE
Contact: <sip:UserB@example.com>
Content-Length: 0



INVITE F5 Proxy-> UA-B

INVITE sip:UserB@example.com SIP/2.0
Via: SIP/2.0/UDP ims.example.com:5060;branch=2
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>
Call-Id: 12345600@here.com
History-Info: <sip:UserA@ims.example.com?Reason=SIP;\ cause=302;
text="Moved Temporarily">; index=1,
<sip:UserB@example.com>;index=2
CSeq: 1 INVITE
Contact: BigGuy <sip:User1@here.com>
Content-Type: application/sdp
Content-Length: <appropriate value>

v=0
o=User1 2890844526 2890844526 IN IP4 client.here.com
s=Session SDP
c=IN IP4 192.0.2.3
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000

180 Ringing F6  UA-B ->Proxy

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP there.com:5060
```

```
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>;tag=5
Call-ID: 12345600@here.com
CSeq: 1 INVITE
Content-Length: 0

180 Ringing F7  Proxy-> UA1

SIP/2.0 180 Ringing
SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>
Call-Id: 12345600@here.com
CSeq: 1 INVITE
Content-Length: 0

/* User B is not available. INVITE is sent multiple
times until it times out. */

  /* The proxy forwards the INVITE to UA-VM after adding the
additional History Information entry. */


INVITE F8  Proxy-> UA-VM

INVITE sip:VM@example.com SIP/2.0
Via: SIP/2.0/UDP ims.example.com:5060;branch=3
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
   To: LittleGuy <sip:UserA@example.com>
Call-Id: 12345600@here.com
History-Info:<sip:UserA@ims.example.com?Reason=SIP;\ cause=302;
text="Moved Temporarily">;index=1,
<sip:UserB@example.com?Reason=SIP;cause=480;\
text="Temporarily Unavailable" >;index=2,
<sip:VM@example.com>;index=3
CSeq: 1 INVITE
Contact: BigGuy <sip:User1@here.com>
Content-Type: application/sdp
Content-Length: <appropriate value>

v=0
o=User1 2890844526 2890844526 IN IP4 client.here.com
s=Session SDP
c=IN IP4 192.0.2.3
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

200 OK F9

SIP/2.0 200 OK UA-VM->Proxy

```
Via: SIP/2.0/UDP ims.example.com:5060;branch=3
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>;tag=3
Call-Id: 12345600@here.com
CSeq: 1 INVITE
Contact: TheVoiceMail <sip:VM@example.com>
Content-Type: application/sdp
Content-Length: <appropriate value>

v=0
o=UserA 2890844527 2890844527 IN IP4 vm.example.com
s=Session SDP
c=IN IP4 192.0.2.4
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

200 OK F10 Proxy->UA1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ims.example.com:5060;branch=3
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:User1@here.com>
To: LittleGuy <sip:UserA@example.com>;tag=3
Call-Id: 12345600@here.com
CSeq: 1 INVITE
Contact: TheVoiceMail <sip:VM@example.com>
Content-Type: application/sdp
Content-Length: <appropriate value>

v=0
o=UserA 2890844527 2890844527 IN IP4 vm.example.com
s=Session SDP
c=IN IP4 192.0.2.4
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

ACK F11 UA1-> UA-VM

```
ACK sip:VM@example.com SIP/2.0
Via: SIP/2.0/UDP here.com:5060
```

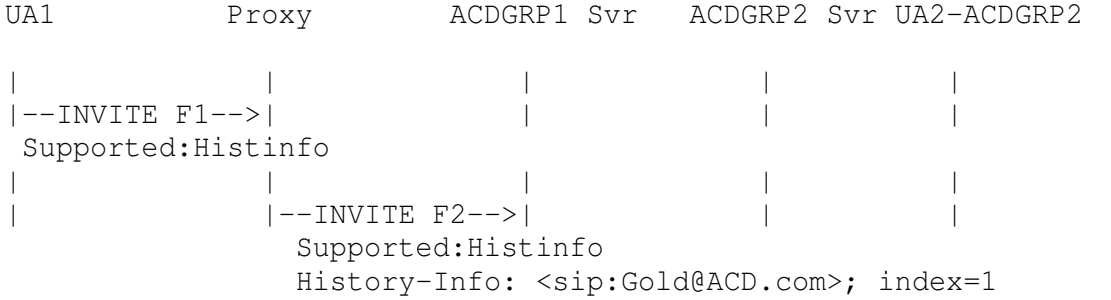
```
From: BigGuy <sip:User1@here.com>
To: LittleGuy<sip:UserA@example.com>;tag=3
Call-Id: 12345600@here.com
CSeq: 1 ACK
Content-Length: 0

/* RTP streams are established between UA1 and
UA-VM. UA-VM starts announcement for UA1 */
```

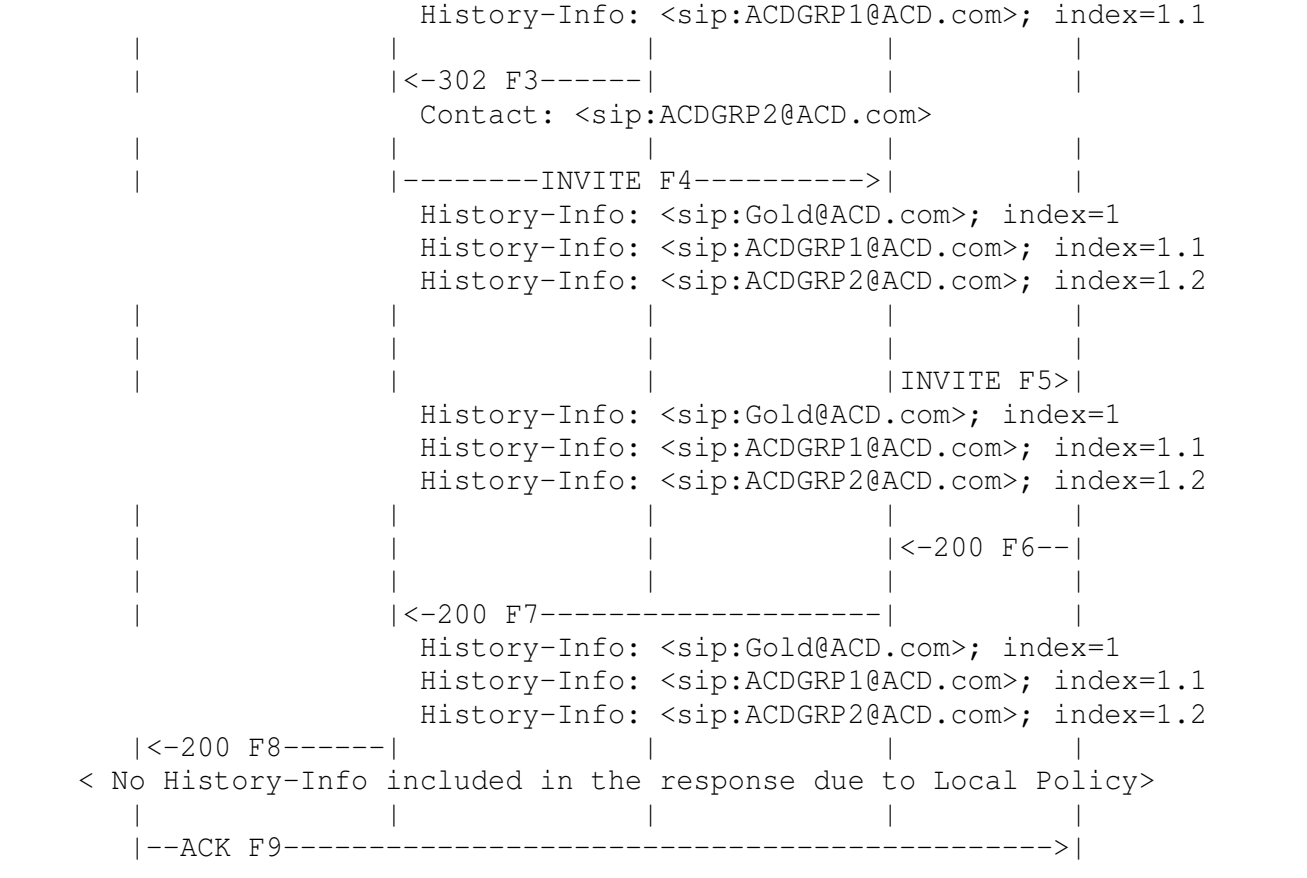
Appendix C Automatic Call Distribution Example

This scenario highlights an example of an Automatic Call Distribution service, where the agents are divided into groups based upon the type of customers they handle. In this example, the Gold customers are given higher priority than Silver customers, so a Gold call would get serviced even if all the agents servicing the Gold group (ACDGRP1) were busy, by retargeting the request to the Silver Group. Upon receipt of the call at the agent assigned to handle the incoming call, based upon the History-Info header in the message, the application at the agent can provide an indication that this is a Gold call, from how many groups it might have overflowed before reaching the agent, etc. and thus can be handled appropriately by the agent.

For scenarios whereby calls might overflow from the Silver to the Gold, clearly the alternate group identification, internal routing or actual agent that handles the call SHOULD not be sent to UA1, thus for this scenario, one would expect that the Proxy would not support the sending of the History-Info in the response, even if requested by the calling UA.

As with the other examples, this is not prescriptive of how one would do this type of service but an example of a subset of processing that might be associated with such a service. In addition, this example is not addressing any aspects of Agent availability, which might also be done via a SIP interface.



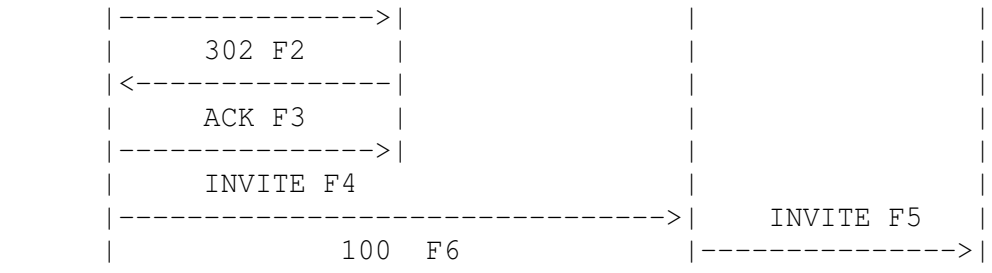


Message Details

[To be completed]

Appendix D Session via Redirect and Proxy Servers

In this scenario, Alice places a call to Bob using first a Redirect server then a Proxy Server. The INVITE message is first sent to the Redirect Server. The Server returns a 302 Moved Temporarily response (F2) containing a Contact header with Bob's current SIP address. Alice then generates a new INVITE with Bob's current SIP address included in another History-Info entry. The INVITE is then sent to Bob via the Proxy Server, with Bob receiving the complete History information; the call then proceeds normally. The complete call flow for this scenario, without the use of History-Info is described in the SIP Basic Call Flow Examples [RFC3665].



Message Details

F1 INVITE Alice -> Redirect Server

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
History-Info: <sip:bob@biloxi.example.com>; index=1
Contact: <sip:alice@client.atlanta.example.com>
Content-Length: 0
```

F2 302 Moved Temporarily Redirect Proxy -> Alice

```
SIP/2.0 302 Moved Temporarily
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
 ;received=192.0.2.1
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=53fHlqlQ2
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
History-Info: <sip:bob@biloxi.example.com>; index=1
Contact: <sip:bob@chicago.example.com;transport=tcp>
Content-Length: 0
```

F3 ACK Alice -> Redirect Server

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=53fHlqlQ2
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
```

```
  CSeq: 1 ACK
  Content-Length: 0


  F4 INVITE Alice -> Proxy 3

  INVITE sip:bob@chicago.example.com SIP/2.0
  Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
  Max-Forwards: 70
  From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
  To: Bob <sip:bob@biloxi.example.com>
  Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
  CSeq: 2 INVITE
  History-Info: <sip:bob@biloxi.example.com?Reason=SIP;cause=302>\
                 text="Moved Temporarily">; index=1,
                <sip:bob@chicago.example.com>; index=2
  Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
  Content-Length: 0


  F5 INVITE Proxy 3 -> Bob

  INVITE sip:bob@client.chicago.example.com SIP/2.0
  Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
  Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
   ;received=192.0.2.1
  Max-Forwards: 69
  Record-Route: <sip:ss3.chicago.example.com;lr>
  From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
  To: Bob <sip:bob@biloxi.example.com>
  Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
  CSeq: 2 INVITE
  History-Info: <sip:bob@biloxi.example.com?Reason=SIP;cause=302>\
                 text="Moved Temporarily">; index=1,
                <sip:bob@chicago.example.com>; index=2,
                <sip:bob@client.chicago.example.com>; index=2.1
  Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
  Content-Length: 0

  Detailed Call Flow continues per section 6.3 in [RFC 3665].
```

Intellectual Property Statement

that it has made any 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 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 which may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr.org.

Disclaimer of Validity

Full Copyright Statement

SIP WG
Internet-Draft
Expires: November 15, 2004

J. Peterson
NeuStar
C. Jennings
Cisco Systems
May 17, 2004

Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)
draft-ietf-sip-identity-02

Status of this Memo

Copyright Notice

Abstract

The existing security mechanisms in the Session Initiation Protocol are inadequate for cryptographically assuring the identity of the end users that originate SIP requests and responses, especially in an interdomain context. This document recommends practices and conventions for identifying end users in SIP messages, and proposes a way to distribute cryptographically secure authenticated identities.

Table of Contents

1. Introduction

This document provides enhancements to the existing mechanisms for authenticated identity management in the Session Initiation Protocol (SIP [1]). An identity, for the purposes of this document, is defined as a canonical SIP address-of-record URI employed to reach a user (such as 'sip:alice@atlanta.com').

RFC3261 enumerates a number of places within a SIP request that a user can express an identity for themselves, notably the user-populated From header field. However, the recipient of a SIP request has no way to verify that the From header field has been populated accurately, in the absence of some sort of cryptographic authentication mechanism.

RFC3261 specifies a number of security mechanisms that can be employed by SIP UAs, including Digest, TLS and S/MIME (implementations may support other security schemes as well). However, few SIP user agents today support the end-user certificates necessary to authenticate themselves via TLS or S/MIME, and furthermore Digest authentication is limited by the fact that the originator and destination must share a pre-arranged secret. It is desirable for SIP user agents to be able to send requests to destinations with they have no previous association - just as in the telephone network today, one can receive a call from someone with whom one has no previous association, and still have a reasonable assurance that their displayed Caller-ID is accurate.

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 RFC2119 [2] and indicate requirement levels for compliant SIP implementations.

3. Background

All RFC3261-compliant SIP user agents support a means of authenticating themselves to a SIP registrar - commonly with a shared secret (Digest authentication, which MUST be supported by SIP user agents, is typically used for this purpose). Registration allows a user agent to express that it is the proper entity to which requests should be sent for a particular address-of-record SIP URI.

Coincidentally, the address-of-record URI of a SIP user is also the URI with which a user commonly populates the From header of requests - in other words, the address-of-record is an identity. So in this

context, users already have a means of providing their identity, which makes good sense: since the contents of a From header field are essentially a 'return address' for SIP requests, being able to prove that you are eligible to receive requests for that 'return address' should be identical to proving that you are authorized to assert this identity.

However, the credentials with which a user agent proves their identity to a registrar cannot be validated by a user agent or proxy server outside your local domain - these credentials are currently only useful for registration. For the purposes of determining whether or not the 'return address' of a request can legitimately be asserted as the identity of the user, SIP entities in other domains require an assurance that the sender of a message is capable of authenticating themselves to a registrar in their own domain.

Ideally, then, SIP user agents should have some way of proving to recipients of SIP messages that their local domain has authenticated them. In the absence of end-user certificates in user agents, it is possible to implement a mediated authentication architecture for SIP in which requests are sent to a server in the user's local domain which authenticates such requests (using the same practices by which the domain would authenticate REGISTER requests). Once a message has been authenticated, the local domain then needs some way to communicate to other SIP entities the sending user has been authenticated. This draft addresses how that imprimatur of authentication can be shared.

RFC3261 already describes an architecture very similar to this in Section 26.3.2.2, in which a user agent authenticates itself to a local proxy server which in turn authenticates itself to a remote proxy server via mutual TLS, creating a two-link chain of transitive authentication between the originator and the remote domain. While this works well in some architectures, there are a few respects in which this is impractical. For one, transitive trust in inherently weaker than an assertion that can be validated end-to-end. It is possible for SIP requests to cross multiple intermediaries in separate administrative domains, in which case transitive trust becomes even less compelling. It also requires intermediaries to act as proxies, rather than redirecting requests to their destinations (redirection lightens loads on SIP intermediaries).

One solution to this problem is to use 'trusted' SIP intermediaries that assert an identity for users in the form of a privileged SIP header. A mechanism for doing so (with the P-Asserted-Identity header) is given in [6]. However, this solution allows only hop-by-hop trust between intermediaries, not end-to-end cryptographic authentication, and it assumes a managed network of nodes with strict

mutual trust relationships, an assumption that is incompatible with widespread Internet deployment.

Accordingly, a new tactic is required for sharing a cryptographic assurance of end-user identity in an intradomain context. Furthermore, this new mechanism must work for both SIP requests and responses. However, there is an additional wrinkle specific to providing identity in a response. While the original address-of-record to which a request is sent is stored in the To header field of the request, it is possible, due to retargeting at intermediaries, it is possible that the request will be forwarded to an entity that has a different AoR (i.e. identity). Since the To header is not changed in responses to a SIP request, the UAC has no way of discovering that new AoR. This is generally known as the "response identity" or "connected party" problem.

4. Requirements

This draft addresses the following requirements:

The mechanism must allow a UAC to provide a strong cryptographic identity assurance to the UAS in a request.

The mechanism must allow a UAS to provide a strong cryptographic identity assurance to the UAC in a response.

User agents that receive identity assurances must be able to validate these assurances without performing any network lookup.

Proxy servers must be capable of adding this identity assurance to requests or responses.

The mechanism must prevent replay of the identity assurance by an attacker.

The mechanism must be capable of protecting the integrity of SIP message bodies (to ensure that media offers and answers are linked to the signaling identity).

It must be possible for a user to have multiple AoRs (i.e. accounts or aliases) under which it is known at a domain, and for the UAC to assert one identity while authenticating itself as another, related, identity, as permitted by the local policy of the domain.

It must be possible, in cases where a request has been retargeted to a different AoR than the one designated in the To header field, for the UAC to ascertain the AoR to which the request has been

sent.

5. Overview of Operations

This section provides an informative (non-normative) high-level overview of the mechanisms described in this document.

Imagine the case where Alice, who has the home proxy of example.com and the address-of-record sip:alice@example.com, wants to communicate with sip:bob@example.org.

Alice generates an INVITE and places her identity in the From header field of the request. She then sends an INVITE over TLS to an authentication service proxy for her domain.

The authentication service authenticates Alice (possibly by sending a Digest authentication challenge) and validates that she is authorized to populate the value of the From header field (which may be Alice's AoR, or it may be some other value that the policy of the proxy server permits her to use). It then computes a hash over some particular headers, including the From header field and the bodies in the message. This hash is signed with the certificate for the domain (example.com, in Alice's case) and inserted in a header field (the new Identity header) in the SIP message. The proxy, as the holder the private key of its domain, is asserting that the originator of this request has been authenticated and that she is authorized to claim the identity (the SIP address-of-record) which appears in the From header field. The proxy also inserts a companion header field that tells Bob how to acquire its certificate, if he doesn't already have it.

When Bob returns a response to the INVITE (such as a 200 OK), a similar set of steps happen. Bob's home proxy asserts his identity in the response. In this instance, the proxy has to insert the header directly into the request - redirection of responses is not possible. When Alice receives the response, she verifies Bob's identity.

If Alice's request for Bob were retargeted, one of two things might happened. If it were retargeted to a domain that was also the responsibility of Bob's home proxy (for example, retargeted from sip:bob@example.com to sip:carol@example.com), then the request would proceed normally and receive an Identity. If Bob's home proxy would retarget the request to some other domain (e.g. sip:bob@example.ORG), then his home proxy would redirect the request rather than proxying it, and Alice would send a new request that could receive a response with an Identity from the new domain.

6. User Agent Behavior: Sending Messages

This mechanism requires one important change to existing user agent behavior for sending requests and responses: user agents using this mechanism to send requests or responses MUST support TLS; moreover, they MUST be capable of establishing a persistent TLS connection with a proxy server that acts as an authentication service. Additionally, there are several practices that should be highlighted in the context of this identity solution.

When a UAC sends a request, it MUST accurately populate the header field that asserts its identity (for a SIP request, this is the From header field). In a request it MUST set the URI portion of its From header to match a SIP, SIPS or TEL URI AoR under which the UAC can register (including anonymous URIs, as described in RFC 3323 [3]). The UAC MUST also be capable of sending requests through an 'outbound' proxy (the authentication service), and of course MUST support the Digest authentication mechanism described in RFC3261. Because this mechanism does not provide integrity protection for the first hop to the authentication service, the UAC MUST send requests to an authentication service only over a TLS connection. Additionally, in order to provide identity for responses, user agents MUST form a persistent TLS connection to a proxy server when a REGISTER is sent.

Since a UAS cannot send a response that does not replicate the contents of the To and From header fields in the corresponding request, UAS response-sending behavior is unchanged. Again, because this mechanism does not provide integrity protection for the first hop of the response path, the UAS SHOULD send responses only over a TLS connection.

7. Authentication Service Behavior

The authentication service authenticates the identity of the message sender and validates that the identity given in the message can legitimately be asserted by the sender. Then it computes a signature over the canonical form of several headers and all the bodies, and inserts this signature into the message.

First, an authentication service MUST extract the identity of the sender. For requests, it inspects the From header field; for responses, the To header field (henceforth the result of this inspection will be referred to as the "identity field). If the identity field contains a SIP or SIPS URI, the authentication service MUST extract the hostname portion of the URI in that header field, and compare this to the domain(s) for which it is responsible. If the identity field uses the TEL URI scheme, the policy of the

authentication service determines whether or not it is responsible for this identity. Some example policies are described in [TODO]. If the authentication service is not responsible for the identity in question, it MAY handle the request as a normal proxy server; see below for more information on authentication service handling of an existing Identity header.

Second, the authentication service must determine whether or not the sender of the request is authorized to claim the identity given in the identity field. In order to do so, the authentication service MUST authenticate the sender of the message. Some possible ways in which this authentication might be performed include:

For requests, challenging the request with a 407 response code using the Digest authentication scheme (or viewing a Proxy-Authentication header sent in the request which was sent in anticipation of a challenge using cached credentials, as described in RFC 3261 Section 22.3)

For requests and responses that are sent over a persistent TLS connection, relying on some prior authentication that was performed at the formation of the connection (most likely, the authentication service previously challenged a REGISTER request sent after the TLS connection was formed, or possibly a prior challenged INVITE that was sent over the TLS connection)

Authorization of the assertion of a particular username in the From header field of a SIP message is a matter of local policy for the authorization service which depends greatly on the manner in which authentication is performed. A RECOMMENDED policy is as follows: the username asserted during Digest authentication MUST correspond exactly to the username in the From header field of the SIP message. However, there are many cases in which a user might manage multiple accounts in the same administrative domain. Accordingly, provided the authentication service is aware of the relationships between these accounts, it might allow a user providing credentials for one account to assert a username associated with another account controlled by the user name. Furthermore, if the AoR asserted in the From header field is anonymous (per RFC3323), then the proxy should authenticate that the user is any valid user in the domain and insert the signature over the From header field as usual.

Third, the authentication service MUST form the identity signature, as described in Section 10, and add an Identity header to the request containing this signature. After the Identity header has been added to the request, the authentication service MUST also add an Identity-Info header. The Identity-Info header contains a URI from which its certificate can be acquired. Details are provided in section Section

10.

Finally, the authentication service MUST forward the message normally.

7.1 UAs acting as an Authentication service

There are some instances in which a user agent may hold the private key of the domain Certificate for its address-of-record. In these cases, the UA MAY perform the services, and add the headers, that the authentication service would normally add.

8. Verifying Identity

When a user agent or proxy server receives a SIP message containing an Identity header, it can inspect the signature to verify the identity of the sender of the message. If an Identity header is not present in a request, and one is required by local policy, then a 428 Use Identity response is sent. If an Identity header is not present in a response, and one is required by local policy, then the recipient of the response MUST communicate this lapse to its user, and MAY immediately terminate any created dialog or ignore transactions, as policy dictates.

In order to verify the identity of the sender of a message, the user agent or proxy server MUST first acquire the certificate for the signing domain. Implementations supporting this specification should have some means of retaining domain certificates (in accordance with normal practices for certificate lifetimes and revocation) in order to prevent themselves from needlessly downloading the same certificate every time a request from the same domain is received. Certificates retained in this manner should be indexed by the URI given in the Identity-Info header field value.

Provided that the domain certificate used to sign this message is unknown, SIP entities discover this certificate by dereferencing the Identity-Info header. The client processes this certificate in the usual ways including checking that it has not expired, that the chain is valid back to a trusted CA, and that it does not appear on revocation lists.

Subsequently, the recipient MUST verify the signature in the Identity header, and compare the identity of the signer (the subjectAltName of the certificate) with the domain portion of the URI in the From header field of the request as described in Section 11. Additionally, the Date, Contact and Call-ID headers MUST be analyzed in the manner described in Section 11; recipients that wish to verify Identity signatures MUST support all of the operations described

there. Any discrepancies or violations MUST be reported to the user.

When the originating user agent of a request receives a response containing an Authenticated Identity Body (AIB, see [4]), it SHOULD compare the identity in the From header field of the AIB of the response with the original value of the To header field in the request. If these represent different identities, the user agent SHOULD render the identity in the AIB of the response to its user. Note that a discrepancy in these identity fields is not necessary an indication of a security breach; normal retargeting may simply have directed the request to a different final destination. Implementers therefore may consider it unnecessary to alert the user of a security violation in this case.

9. Proxy Server Behavior

In most respects, a proxy server behaves normally when it receives a SIP request or response containing an Identity header. This mechanism is fully backwards-compatible with existing RFC3261 proxy behavior. However, if a proxy intends to act as an authentication service for responses to requests it receives, it must exhibit some additional behavior to ensure that retargeting requests are handled properly. Essentially, a proxy server MUST NOT provide an Identity header for a request that it retargets to a different administrative domain. It is the responsibility of that administrative domain to provide its own identity assertion, if it can. However, proxying the request to a remote domain where identity services may be provided has its own problems - the originator of the request has no way to know whether the request was legitimately retargeted, or if any responses it receives from the new domain are spoofed or otherwise illegitimate. It is thus much more secure for the proxy server to redirect in cases where it might otherwise retarget.

If a proxy server intends to act as an authentication service for a response to a SIP request that it is forwarding, it MUST do ALL of the following:

Ascertain if it is responsible for the domain indicated in the Request-URI field of the request. If not, it MUST forward the request normally. If so, it must then:

Determine the route set of targets to which this request might be forwarded. From that target list, the proxy must determine which contact addresses are associated with persistent TLS connections that have been established to the proxy server. It places all such targets (if any) into a primary route set for the call, and places remaining targets into a secondary route set for the call. It performs this operations irrespective of any qvalues associated

with the contact addresses.

The proxy then MUST follow normal administrative policies for forwarding the request to any targets in the primary route set (which may involve qvalue calculations or any other behaviors described in RFC3261). Before the proxy forwards any responses to this request upstream, the proxy server MUST act as an authentication service (as described in Section 7), adding an Identity and Identity-Info header.

If there are no appropriate responses from the primary route set for the proxy server to forward upstream, it moves on to the secondary route set (essentially, the proxy server forks sequentially, exploring the primary route set as one cluster, and then moves on to the secondary set). The proxy server is unable to act as an authentication service for those contact addresses. Accordingly, the proxy server MUST NOT explore these route targets itself; instead, it MUST redirect the request with a 3xx class response containing the contact addresses that constitute the secondary route set.

In order to build the primary route set for the call, the location service associated with the domain of the proxy server MUST implement additional intelligence to determine which contact addresses are associated with a persistent TLS connection - this is used to determine when the server should act as a proxy and when it should redirect. When the SIP registrar receives a REGISTER request over a persistent TLS connection, it MUST compare any contact addresses appearing in Contact header fields to the topmost Via header field in the REGISTER request. If the host portion of a contact address matches the hostname given in the topmost Via header field, then that contact address is said to be "associated" with the persistent TLS connection over which the REGISTER was received. Location services must mark or flag these contact addresses accordingly, and remember the identity that the user provided when they were authenticated during registration. Only these contact addresses are added to the primary route set by a proxy server that wishes to act as an authentication service for responses.

Additionally, domain policy may require proxy servers to inspect and verify the identity provided in SIP requests. The proxy server may wish to ascertain the identity of the sender of the message to provide spam prevention or call control services. Even if a proxy server does not act as an authentication service, it MAY verify the signature present in an Identity header before it makes a forwarding decision for a request. Proxy servers MUST NOT remove or modify the Identity or Identity-Info headers.

10. Header Syntax

This document specifies two new SIP headers: Identity and Identity-Info. Each of these headers can appear only once in a SIP message.

```
Identity = "Identity" HCOLON signed-identity-digest
signed-identity-digest = LDQUOT 32LHEX RDQUOT

Identity-Info = "Identity-Info" HCOLON ident-info
Ident-info = LAQUOT absoluteURI RAQUOT
```

To create the contents of the signed-identity-digest, the following elements of a SIP message MUST placed in the string in the order specified here, separated by a colon:

The AoR of the UA sending the message, or the 'identity field'. For a request, this is the addr-spec from the From header field; for responses, the addr-spec of the To header field. This needs to be in lower case and to be represented as a SIP URI.

The callid from Call-Id header field.

The Date header field, with exactly one space each for each SP and the weekday and month items case set as shown in BNF in 3261. The first letter is upper case and the rest of the letters are lower case.

The addr-spec component of the Contact header field value.

The body content of the message with the bits exactly as they are in the message.

For more information on the security properties of these headers, and why their inclusion mitigates replay attacks, see [4]. The precise formulation of this digest-string is, therefore:

```
digest-string = addr-spec HCOLON callid HCOLON SIP-Date
             HCOLON addr-spec HCOLON message-body
```

Note again that the first addr-spec MUST be taken from the From header field value, and the second addr-spec from the Contact header field value.

After the digest-string is formed, it MUST be hashed and signed with the certificate for the domain, as follows: compute the results of signing this string with sha1WithRSAEncryption as described in RFC 3370 and base64 encode the results as specified in RFC 3548. Put the result in the Identity header.

Note on this choice: Assuming a 1024 bit RSA key, the raw signature will result in about 170 octets of base64 encoded data. For comparison's sake, a typical HTTP Digest Authorization header (such as those used in RFC3261) with no cnonce is around 180 octets. From a speed point of view, a 2.8GHz Intel processor does somewhere in the range of 250 RSA 1024 bits signs per second or 1200 RSA 512 bits signs; verifies are roughly 10 times faster. Hardware accelerator cards are available that speed this up.

The Identity-Info header MUST contain either an HTTPS URI or a SIPS URI. If it contains an HTTPS URI, the URI must dereference to a resource that contains a single MIME body containing the certificate of the authentication service. If it is a SIPS URI, then the authentication service intends for a user agent that wishes to fetch the certificate to form a TLS connection to that URI, acquire the certificate during normal TLS negotiation, and close the connection.

This document adds the following entries to Table 2 of [1]:

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Identity		a	-	o	-	o	o	-

	SUB	NOT	REF	INF	UPD	PRA
	o	o	o	-	-	-

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Identity-Info		a	-	o	-	o	o	-

	SUB	NOT	REF	INF	UPD	PRA
	o	o	o	-	-	-

11. Security Considerations

This document describes a mechanism which provides a signature over the Contact, Date, Call-ID, and 'identity fields' (addr-spec of the From header field for requests, and To header field for responses) of SIP messages. While a signature over the identity field alone would be sufficient to secure a URI alone, the additional headers provide replay protection and reference integrity necessary to make sure that the Identity header will not be used in cut-and-paste attacks. In general, the considerations related to the security of these headers are the same as those given in RFC3261 for including headers in tunneled 'message/sip' MIME bodies (see Section 23 in particular).

The identity field indicates the identity of the sender of the message. The Date and Contact headers provide reference integrity and replay protection, as described in RFC3261 Section 23.4.2. Implementations of this specification MUST NOT consider valid a request with an outdated Date header field (the RECOMMENDED interval is that the Date header must indicate a time within 3600 seconds of the receipt of a message). Implementations MUST also record Call-IDs received in valid requests containing an Identity header, and MUST remember those Call-IDs for at least the duration of a single Date interval (i.e. 3600 seconds). Accordingly, if an Identity header is replayed within the Date interval, receivers will recognize that it is invalid because of a Call-ID duplication; if an Identity header is replayed after the Date interval, receivers will recognize that it is invalid because the Date is stale. The Contact header field is included to tie the Identity header to a particular device instance that generated the request. Were an active attacker to intercept a request containing an Identity header, and cut-and-paste the Identity header field into their own request (reusing the identity fields, Contact, Date and Call-ID fields that appear in the original message), they would not be eligible to receive SIP requests from the called user agent, since those requests are routed to the URI identified in the Contact header field.

This mechanism also provides a signature over the bodies of SIP requests. The most important reason for doing so is to protect SDP bodies carried in SIP requests. There is little purpose in establishing the identity of the user agent that provided the signaling if a man-in-the-middle can change the SDP and direct media to an alternate address. Note however that this is not perfect end-to-end security. The authentication service itself, when instantiated at a intermediary, can change the SDP (and SIP headers, for that matter) before providing a signature. Thus, while this mechanism reduces the chance that a man-in-the-middle will interfere with sessions, it does not eliminate it entirely. Since it is assumed that the user trusts their local domain to vouch for their security, they must also trust the service not to violate the integrity of their message bodies without good reason.

Users SHOULD NOT provide credentials to an authentication service to which they cannot initiate a direct connection, preferably one secured by TLS. If a user does not receive a certificate from the authentication service over this TLS that corresponds to the expected domain (especially when they receive a challenge via a mechanism such as Digest), then it is possible that a rogue server is attempting to pose as a authentication service for a domain that it does not control, possibly in an attempt to collect shared secrets for that domain. If a user cannot connect directly to the desired authentication service, the user SHOULD at least use a SIPS URI to

ensure that mutual TLS authentication will be used to reach the remote server.

Relying on an Identity header generated by a remote administrative domain assumes that the issuing domain uses some trustworthy practice to authenticate its users. However, it is possible that some domains will implement policies that effectively make users unaccountable (such as accepting unauthenticated registrations from arbitrary users). The value of an Identity header for such domains is questionable.

Since a domain certificate is used by an authentication service (rather than individual certificates for each identity), certain problems can arise with name subordination. For example, if an authentication service holds a common certificate for the hostname 'sip.atlanta.com', can it legitimately sign a token containing an identity of 'sip:alice@atlanta.com'? It is difficult for the recipient of a request to ascertain whether or not 'sip.atlanta.com' is authoritative for the 'atlanta.com' domain unless the recipient has some foreknowledge of the administration of 'atlanta.com'. Therefore, it is RECOMMENDED that user agent recipients of authentication tokens notify end users if there is ANY discrepancy between the subjectAltName of the signers certificate and the identity within the authentication token. Minor discrepancies MAY be characterized as such. Additionally, relying parties MAY follow the procedures in RFC3264 to look up on the domain portion of the identity in the From header field in the DNS, and compare the SIP services listed for that domain with the subjectAltName of the certificate; this can give the relying party a better sense of the canonical SIP services for that domain.

Because the domain certificates that can be used by authentication services need to assert only the hostname of the authentication service, existing certificate authorities can provide adequate certificates for this mechanism. However, not all proxy servers and user agents will be able support the root certificates of all certificate authorities, and moreover there are some significant differences in the policies by which certificate authorities issue their certificates. This document makes no recommendations for the usage of particular certificate authorities, nor does it describe any particular policies that certificate authorities should follow, but it is anticipated that operational experience will create de facto standards for the purposes of authentication services. Some federations of service providers, for example, might only trust certificates that have been provided by a certificate authority operated by the federation.

12. IANA Considerations

This document specifies two new SIP headers: Identity and Identity-Info. Their syntax is given in Section 10. This document requests that IANA add these headers to the SIP header registry.

This document also defines a new SIP response code, 428 "Use Identity", as described in Section 8.

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", RFC 2119, March 1997.

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

[4] Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-01 (work in progress), October 2002.

Informative References

[5] Kohl, J. and C. Neumann, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[6] 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.

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

[8] Olson, S., "A Mechanism for Content Indirection in SIP Messages", draft-ietf-sip-content-indirect-mech-01 (work in progress), August 2002.

[9] Freed, N., "Definition of the URL MIME External-Body Access-Type", RFC 2017, November 1996.

Authors' Addresses

Jon Peterson
NeuStar, Inc.
1800 Sutter St
Suite 570
Concord, CA 94520
US

Phone: +1 925/363-8720
EMail: jon.peterson@neustar.biz
URI: http://www.neustar.biz/

Cullen Jennings
Cisco Systems
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 902-3341
EMail: fluffy@cisco.com

Appendix A. Acknowledgments

The authors would like to thank Eric Rescorla, Rohan Mahy, Robert Sparks, Jonathan Rosenberg, Mark Watson and Patrik Faltstrom for their comments.

Appendix B. Changelog

Changes from draft-ietf-sip-identity-01:

- Completely changed underlying mechanism - instead of using an AIB, the mechanism now recommends the use of the Identity header and Identity-Info header

- Numerous other changes resulting from the above

- Various other editorial corrections

Changes from draft-peterson-sip-identity-01:

- Split off child draft-ietf-sip-authid-body-00 for defining of the AIB

- Clarified scope in introduction

- Removed a lot of text that was redundant with RFC3261 (especially about authentication practices)

- Added mention of content indirection mechanism for adding token to requests and responses

- Improved Security Considerations (added piece about credential strength)

Changes from draft-peterson-sip-identity-00:

- Added a section on authenticated identities in responses

- Removed hostname convention for authentication services

- Added text about using 'message/sip' or 'message/sipfrag' in authenticated identity bodies, also RECOMMENDED a few more headers in sipfrags to increase reference integrity

- Various other editorial corrections

Full Copyright Statement

Acknowledgement

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

Network Working Group
Internet-Draft
Expires: September 18, 2004

H. Schulzrinne
Columbia U.
J. Polk
Cisco
March 20, 2004

Communications Resource Priority for the Session Initiation Protocol (SIP)
draft-ietf-sip-resource-priority-03

Status of this Memo

Copyright Notice

Abstract

This document defines two new SIP header fields for communications resource priority, namely "Resource-Priority" and "Accept-Resource-Priority". The "Resource-Priority" header field can influence the behavior of SIP UAs, such as GSTN gateways, and SIP proxies. It does not directly influence the forwarding behavior of IP routers.

Table of Contents

1. Introduction

During emergencies, communications resources including telephone circuits, IP bandwidth and gateways between the circuit-switched and IP networks may become congested. Congestion can occur due to heavy usage, loss of resources caused by the natural or man-made disaster and attacks on the network during man-made emergencies. This congestion may make it difficult for persons charged with emergency assistance, recovery or law enforcement to coordinate their efforts. As IP networks become part of converged or hybrid networks along with public and private circuit-switched (telephone) networks, it becomes necessary to ensure that these networks can assist during such emergencies.

Also, users may want to interrupt their lower-priority communications activities and dedicate their end system resources to the high-priority communications attempt if a high-priority communications request arrives at their end system.

There are many IP-based services that can assist during emergencies. This memo only covers real-time communications applications involving SIP, including voice-over-IP, multimedia conferencing and instant messaging/presence.

Session Initiation Protocol (SIP) [RFC3261] applications may involve at least five different resources that may become scarce and congested during emergencies. These resources include gateway resources, circuit-switched network resources, IP network resources, receiving end system resources and SIP proxy resources. IP network resources are beyond the scope of SIP signaling and are therefore not considered here.

In order to improve emergency response, it may become necessary to prioritize access to SIP-signaled resources during periods of emergency-induced resource scarcity. We call this "resource prioritization".

Currently, SIP does not include a mechanism that allows a request originator to indicate to a SIP element that it wishes the request to invoke such resource prioritization. To address this need, this document adds a SIP protocol element that labels certain SIP requests.

This document defines (Section 3) a new SIP [RFC3261] header field for communications resource priority, called 'Resource-Priority' This header field MAY be used by SIP user agents, including GSTN gateways and terminals, and SIP proxy servers to influence their treatment of SIP requests, including the priority afforded to GSTN calls. For

GSTN gateways, the behavior translates into analogous schemes in the GSTN, for example the ITU Recommendation Q.735.3 [Q.735.3] prioritization mechanism, in both the GSTN-to-IP and IP-to-GSTN directions.

The 'Resource-Priority' header field may be used in several situations. A SIP request with such an indication can be treated differently in these situations:

1. The request can be given elevated priority for access to GSTN gateway resources such as trunk circuits.
2. The request can interrupt lower-priority requests at a user terminal, such as an IP phone.
3. The request can carry information from one multi-level priority domain in the telephone network, e.g., using the facilities of Q.735.3 [Q.735.3], to another, without the SIP proxies themselves inspecting or modifying the header field.
4. In SIP proxies and back-to-back user agents, requests of higher priorities may displace existing signaling requests or bypass GSTN gateway capacity limits in effect for lower priorities.

This header field is related to, but differs in semantics from, the 'Priority' header field (RFC 3261 [RFC3261], Section 20.26). The 'Priority' header field describes the importance that the SIP request should have to the receiving human or its agent. For example, that header may be factored into decisions about call routing to mobile devices and assistants and call acceptance when the call destination is busy. The 'Priority' header field does not affect the usage of PSTN gateway or proxy resources, for example. In addition, any UAC can assert any 'Priority' value, while access to resource priority values is subject to authorization.

While the 'Resource-Priority' header does not directly influence the forwarding behavior of IP routers or the use of communications resources such as packet forwarding priority, procedures for using this header to cause such influence may be defined in other documents.

Existing implementations of RFC 3261 that do not participate in the resource priority mechanism follow the normal rules of RFC 3261, Section 8.2.2: "If a UAS does not understand a header field in a request (that is, the header field is not defined in this specification or in any supported extension), the server MUST ignore that header field and continue processing the message." Thus, the use of this mechanism is wholly invisible to existing implementations unless the request includes the Require header field with the Resource-Priority option flag.

The mechanism described here can be used for emergency preparedness in emergency telecommunications systems, but is only a small part of an emergency preparedness network and is not restricted to such use.

The mechanism is structured so that it works in all SIP/RTP transparent networks defined in [RFC3487], i.e., all network elements and SIP proxies let valid SIP requests pass through unchanged. This is important since it is likely that this mechanism will often be deployed in networks where the edge networks are unaware of the resource priority mechanism and provide no special privileges to such requests. The request then reaches a PSTN gateway or set of SIP elements that are aware of the mechanism.

For conciseness, we refer to SIP proxies and user agents that act on the 'Resource-Priority' header field as RP actors.

We define the header field syntax in Section 3 and then describe the behavior of user agents and proxies in Section 4.3 through Section 4.5. Section 6 briefly describes how this feature affects existing systems that do not support it. Third-party authentication is discussed in Section 5, while general security issues are enumerated in Section 8. This specification does not propose any new SIP security mechanisms. Examples can be found in Section 7.

The mechanism aims to satisfy the requirements in [RFC3487].

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 BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. The Resource-Priority and Accept-Resource-Priority SIP Header Fields

This document defines the 'Resource-Priority' and 'Accept-Resource-Priority' SIP header fields.

The SIP element behavior is described for UACs in Section 4.3, for UAS in Section 4.4, for proxies in Section 4.5.

3.1 The Resource-Priority Header Field

The 'Resource-Priority' header field marks a SIP request as desiring prioritized resource access, as described in the introduction. In responses, the 'Resource-Priority' header fields indicates the actual resource priority that was granted to the request. While it is

usually the same value contained in the request, implementations MAY insert a different value based on local policy.

There is no requirement that all requests within a SIP dialog or call use the 'Resource-Priority' header field.

The syntax of the 'Resource-Priority' header field is as follows:

```
Resource-Priority  = "Resource-Priority" HCOLON
                     Resource-value *(COMMA Resource-value)
Resource-value     = namespace "." r-priority
namespace          = *(alphanum / "-")
r-priority         = *(alphanum / "-")
```

An example 'Resource-Priority' header field is shown below:

```
Resource-Priority: q735.1, dsn.flash
```

The 'Resource-value' parameter in the 'Resource-Priority' header indicates the resource priority desired by the request originator. Since a request may traverse multiple administrative domains with multiple different namespaces, it is necessary to be able to enumerate several different namespaces. However, each namespace MUST NOT appear more than once in a SIP message.

Each resource value is formatted as 'namespace' '.' 'priority value'. The value is drawn from the namespace identified by the 'namespace' token. Namespaces and priorities are case-independent ASCII. Each namespace has at least one priority value. Namespaces and priority values within each namespace are registered with IANA (Section 9); some initial namespaces are described in Section 9.5.

There may be multiple resource values or, equivalently, multiple 'Resource-Priority' header field instances.

3.2 The Accept-Resource-Priority Header Field

The 'Accept-Resource-Priority' response header field enumerates the resource values a SIP user agent server implements. The syntax of the 'Accept-Resource-Priority' header field is as follows:

```
Accept-Resource-Priority = "Accept-Resource-Priority" HCOLON
                           [Resource-value *(COMMA Resource-value)]
```

An example is given below:

```
Accept-Resource-Priority: dsn.flash-override,
  dsn.flash, dsn.immediate, dsn.priority, dsn.routine
```

3.3 Usage of the Resource-Priority and Accept-Resource-Priority Header Fields

Header field	where	proxy	INV	ACK	CAN	BYE	REG	OPT	PRA
Resource-Priority	R	amd	o	o	o	o	o	o	o
Resource-Priority	200	-	o	-	o	o	o	o	o
Accept-Resource-Priority	200	-	o	o	o	o	o	o	o
Accept-Resource-Priority	417	-	m	-	m	m	m	m	m
Accept-Resource-Priority	420	-	o	-	o	o	o	o	o

Header field	where	proxy	SUB	NOT	UPD	MSG	REF	INF	PUB
Resource-Priority	R	amd	o	o	o	o	o	o	o
Resource-Priority	200	-	o	o	o	o	o	o	o
Accept-Resource-Priority	200	-	o	o	o	o	o	o	o
Accept-Resource-Priority	417	-	m	m	m	m	m	m	m
Accept-Resource-Priority	420	-	o	o	o	o	o	o	o

Other request methods MAY define their own handling rules; unless otherwise specified, recipients MAY ignore these header fields. 'Accept-Resource-Priority' MUST be returned in 420 (Not Supported) responses marked as 'o' in table above if the element implements the resource priority mechanism with some other namespaces or priority values, but does not implement the particular namespace or priority value contained in the request.

3.4 The Resource-Priority Option Tag

This document also defines the "resource-priority" option tag. The behavior is described in Section 4.2.2 and the IANA registration is in Section 9.2.

4. Behavior of SIP Elements that Receive Prioritized Requests

4.1 General Rules

All user agent servers and proxy servers that receive SIP requests share certain common behavior, which we describe below. Behavior that is specific to user agent servers is covered in Section 4.4, while Section 4.5 deals with proxy behavior.

A SIP element supporting this specification MUST be able to interpret the 'Resource-Priority' header field in INVITE, ACK, PRACK [RFC3262], MESSAGE [RFC3428], UPDATE [RFC3311], SUBSCRIBE [RFC3265] and NOTIFY [RFC3265] requests, if it supports a particular request. (This does not imply that all elements supporting this specification need to support all of these request methods.) In all such requests, the

priority MAY influence the order in which requests are handled and MUST influence the resources, such as circuits, bandwidth or memory, allocated based on the request. For example, for SUBSCRIBE, a higher-priority request may get preferential treatment if storage or bandwidth for notifications are scarce, possibly displacing a lower-priority subscription. (As always, the precise behavior is defined by a namespace definition, or, if left unspecified, by an implementation or configuration.)

A SIP element MAY ignore the header field in other requests unless the request definition defines behavior for the particular method.

If a request contains multiple valid namespace/priority values, the request is treated according to the highest supported and authorized value. The total ordering of priorities between different namespaces is defined by local policy.

The OPTIONS request can be used to determine if an element supports the mechanism. A compliant implementation MUST return a 'Accept-Resource-Priority' header field in OPTIONS responses enumerating all valid resource values. An implementation MAY reveal this capability only to authorized UACs. (Note that an overloaded UAS may not be able to provide this information at all times.) Note that according to RFC 3261, proxies reached with a Max-Forwards value of zero answer the OPTIONS request, allowing a UAC to discover the capabilities of both proxies and the UAS.

4.2 Error Conditions

4.2.1 Known Namespace and Priority Value

Two error conditions can occur if a request reaches an element that supports the namespace and resource priority. Elements receiving requests with namespaces or priority values that they do not understand act according to the rules in the next section.

Insufficient authorization: If the element receives a request with a namespace and priority value it recognizes, but the originator is not authorized for that level of service, the element MUST return a 403 (Forbidden) response.

Insufficient resources: If there are insufficient resources at an element for a given priority, a request might be delayed or refused, depending on local policy or the definition of the namespace. If it is refused, the element returns a 503 (Service Unavailable) response. The response MAY also include a 'Warning' header with warning code 370 (Insufficient Bandwidth) if the request failed due to insufficient capacity for the media streams, rather than insufficient signaling capacity.

The 503 (Service Unavailable) response provides sufficient indication to the originator to re-attempt with a higher appropriate resource priority or to add a resource priority indication, if authorized.

4.2.2 Handling Unknown Namespaces and Priority Values

When handling requests with unknown namespsaces or priority values, elements can operate in two modes, "strict" and "loose". If the request includes a 'Require' header field with the 'Resource-Priority' option tag, a UAS MUST follow the strict-mode rules, otherwise UAS and proxies may choose either mode according to local policy.

Following standard SIP behavior (Section 8.2.2.3 of [RFC3261]), a UAS operating in strict mode MUST reject the request with response code 420 (Bad Extension) if it does not understand the resource priority mechanisms such as the 'Resource-Priority' header field.

For example, a gateway that is unaware of a resource priority namespace might accept a request at non-elevated priority, but then the request could later be preempted by other requests. Also, use of the 'Require' restriction ensures that in parallel forking, only branches that support the resource priority mechanism succeed.

The use of the 'Resource-Priority' option tag with 'Proxy-Require' is NOT RECOMMENDED.

4.2.2.1 Strict Mode

In strict mode, an element that receives a request with a 'Resource-Priority' header field containing one or more namespace or priority values that it does not implement rejects the request with status code 417 (Unknown Resource-Priority) and includes a 'Accept-Resource-Priority' header field enumerating all the resource values that the server is willing to process. Note that the user may not be authorized to use all of these resource values.

Strict mode is particularly useful for operational testing of systems supporting resource priority, as otherwise it might be difficult to detect under non-overload conditions whether an element supports the functionality or not.

4.2.2.2 Loose Mode

In loose mode, unknown priority values or namespaces are ignored; the request is treated as if these values were not included. If there are no valid priority values or namespaces, the request is treated as if it had no 'Resource-Priority' header field. Thus, no 417 (Unknown

Resource-Priority) is generated.

4.3 User Agent Client Behavior

SIP UACs supporting this specification MUST be able to generate the 'Resource-Priority' header field for requests that require elevated resource access priority.

If the request is returned with 417 (Unknown Resource-Priority), the UAC MAY retry the request with a different set of namespace/priority combinations, drawing from the values returned by the 'Accept-Resource-Priority' header field in the response.

4.4 User Agent Server Behavior

If the UAS understands the resource value, but refuses to honor the request with elevated priority for this particular user, it returns the 403 (Forbidden) response code. It MAY include the list of resource values that the user is allowed to use in the 'Accept-Resource-Priority' response header field.

> The lookup of the authorized values may take significant resources since it may involve an AAA interaction. Thus, it seems imprudent to require that the list is customized to the user. In general, legitimate users know their highest resource value that they are entitled to.

The precise effect of the 'Resource-Priority' indication depends on the type of UAS, the namespace and local policy. For example, a circuit-switched telephony gateway might move requests with elevated priority to the front of the queue of requests waiting for outbound lines, it may utilize additional resources or it may preempt existing calls. For a terminal, such as a SIP phone, requests with elevated priority might trigger a special alert tone or preempt other, lower-priority ongoing calls. The generic protocol mechanism described here does not mandate the particular element behavior, but namespace definitions, such as the ones in Section 9.5, need to spell out the desired behavioral properties of user agents and proxy servers.

4.5 Proxy Behavior

SIP proxies MAY ignore, inspect, insert and modify the 'Resource-Priority' header field. SIP proxies MAY downgrade the 'Resource-Priority' of a request or reject unauthenticated requests. If there are multiple namespace or priority choices available to the user agent, a proxy MAY return the request with an appropriate 'Accept-Resource-Priority' header field. Details are a matter of

local policy.

This behavior is similar to that for any header field, as a UA can decide to reject a request for the presence, absence or value of any information in the request. The session policy mechanism does not fit well, as user agents may not have a choice in the namespace or priority available to them, there are no privacy concerns and the resource priority mechanism does not involve message bodies or session descriptions.

If a stateful proxy has authorized a particular resource priority level and if it offers differentiated treatement to responses containing resource priority levels, the proxy SHOULD ignore any higher value contained in responses, to avoid that colluding user agents artificially raise the priority level.

It is unlikely that the resource priority value in responses will have any influence on response handling.

A SIP proxy MAY use the 'Resource-Priority' indication in its routing decisions, e.g., to find a next hop that is reserved for a particular resource priority.

There do not appear to be any special considerations when forking requests containing a resource priority indication.

Otherwise, the proxy behavior is the same as for user agent servers Section 4.4).

5. Third-Party Authentication

In some case, the RP actor may not be able to authenticate the requestor or determine whether an authenticated user is authorized to make such a request. In these circumstances, the SIP entity may avail itself of general SIP mechanisms that are not specific to this application. The authenticated identity management mechanism [I-D.ietf-sip-authid-body] allows a third party to verify the identity of the requestor and certify this towards an RP actor. In networks with mutual trust, the SIP asserted identity mechanism [RFC3325] can help the RP actor determine the identity of the requestor.

6. Backwards Compatibility

The resource priority mechanism described in this document is fully backwards compatible with SIP systems following RFC 3261 [RFC3261]. Systems that do not understand the mechanism can only deliver standard, not elevated, service priority. User agent servers and

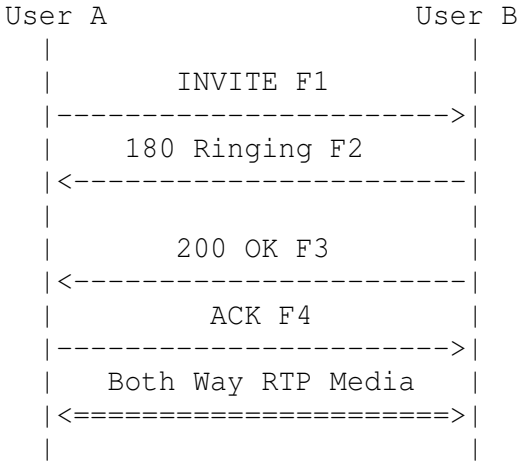
proxies can ignore any 'Resource-Priority' header field just like any other unknown header field and then treat the request like any other request. Naturally, the request may still succeed.

Introducing 'Require' or 'Proxy-Require' would not help, as systems that do not support the mechanism will not improve by rejecting the request due to feature failure. Since the intent of resource priority indications is to increase the probability of call completion, adding failure modes appears counterproductive.

7. Examples

The SDP message body and the BYE and ACK exchanges are the same as in RFC 3665 [RFC3665] and omitted for brevity.

7.1 Simple Call



In this scenario, User A completes a call to User B directly. The call from A to B is marked with a resource priority indication.

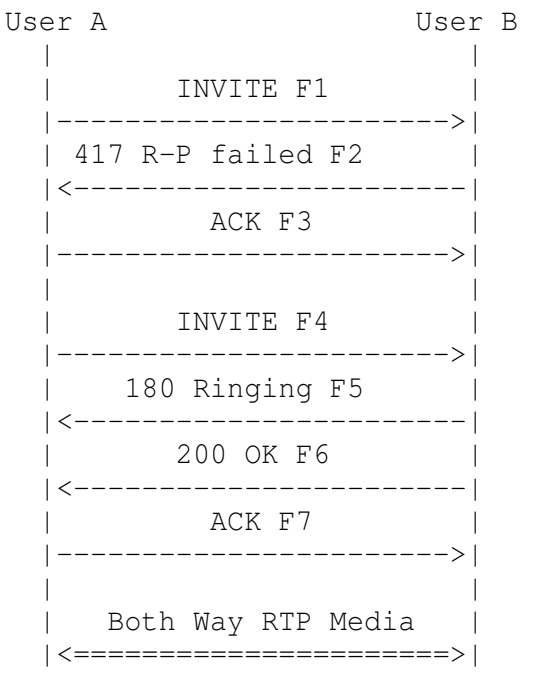
F1 INVITE User A -> User B

```
INVITE sip:UserB@biloxi.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 INVITE
Resource-Priority: dsn.flash
Contact: <sip:UserA@client.atlanta.com;transport=tcp>
Content-Type: application/sdp
Content-Length: ...
```

```
...

F2 180 Ringing User B -> User A

SIP/2.0 180 Ringing
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bf9
  ;received=192.0.2.101
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 INVITE
Resource-Priority: dsn.flash
Contact: <sip:UserB@client.biloxi.com;transport=tcp>
Content-Length: 0


F3 200 OK User B -> User A

SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bf9
  ;received=192.0.2.101
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 INVITE
Resource-Priority: dsn.flash
Contact: <sip:UserB@client.biloxi.com;transport=tcp>
Content-Type: application/sdp
Content-Length: ...

...
```

7.2 Receiver Does Not Understand Namespace

In this example, the receiving UA does not understand the "dsn" namespace and thus returns a 417 (Unknown Resource-Priority) status code. We omit the message details for messages F5 through F7 since they are essentially the same as in the first example.



F1 INVITE User A -> User B

```
INVITE sip:UserB@biloxi.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 INVITE
Resource-Priority: dsn.flash
Contact: <sip:UserA@client.atlanta.com;transport=tcp>

Content-Type: application/sdp
Content-Length: ...

...
```

F2 417 Resource-Priority failed User B -> User A

```
SIP/2.0 417 Resource-Priority failed
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bf9
  ;received=192.0.2.101
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 INVITE
```

```
Accept-Resource-Priority: q735.0, q735.1, q735.2, q735.3, q735.4
Contact: <sip:UserB@client.biloxi.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 0

F3 ACK User A -> User B

ACK sip:UserB@biloxi.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bd5
Max-Forwards: 70
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 ACK
Content-Length: 0

F4 INVITE User A -> User B

INVITE sip:UserB@biloxi.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>
Call-ID: 3848276298220188511@atlanta.com
CSeq: 2 INVITE
Resource-Priority: q735.3
Contact: <sip:UserA@client.atlanta.com;transport=tcp>

Content-Type: application/sdp
Content-Length: ...
...
```

8. Security Considerations

Any resource priority mechanism can be abused to obtain resources and thus deny service to other users. An adversary may be able to take over a particular gateway, cause additional congestion during PSTN during emergencies or deny service to legitimate users.

While the indication itself does not have to provide separate authentication, any SIP request carrying such information has higher authentication requirements than regular requests. Below, we describe authentication and authorization aspects, confidentiality and privacy requirements, protection against denial of service attacks and anonymity requirements. Naturally, the general discussion in RFC 3261 [RFC3261] applies.

8.1 Authentication and Authorization

Prioritized access to network and end system resources imposes particularly stringent requirements on authentication and authorization mechanisms since access to prioritized resources may impact overall system stability and performance, not just result in theft of, say, a single phone call.

Under certain emergency conditions, the network infrastructure, including its authentication and authorization mechanism, may be under attack.

Given the urgency during emergency events, normal statistical fraud detection may be less effective, thus placing a premium on reliable authentication.

Common requirements for authentication mechanisms apply, such as resistance to replay, cut-and-paste and bid-down attacks.

Authentication MAY be SIP-based or use other mechanisms. Use of Digest authentication and/or S/MIME is RECOMMENDED for UAS authentication. Digest authentication requires that the parties share a common secret, thus limiting its use across administrative domains. SIP systems employing resource priority SHOULD implement S/MIME at least for integrity, as described in Section 23 of [RFC3261]. However, in some environments, asserted identity [RFC3325] and transitive trust may be used to build a sufficiently robust system. Section 5 describes third-party authentication.

Trait-based authorization [I-D.ietf-sipping-trait-authz] "entails an assertion by a authorization service of attributes associated with an identity" and may be appropriate for this application as it avoids that all network elements need to maintain or consult a mapping from user identifiers to authorizations.

Authorization may be based on factors beyond the identity of the caller, such as the requested destination. Namespaces MAY also impose particular authentication or authorization consideration that are stricter than the baseline described here.

8.2 Confidentiality and Integrity

Calls which use elevated resource priority levels provided by the 'Resource-Priority' header field are likely to be sensitive and often need to be protected from intercept and alteration. In particular, requirements for protecting the confidentiality of communications relationships may be higher than for normal commercial service. For SIP, the 'To', 'From', 'Organization', 'Subject' and 'Via' header

fields are examples of particularly sensitive information. Systems MUST implement encryption at the transport level using TLS and MAY implement other transport-layer or network-layer security mechanisms. UACs SHOULD use the "sips" URI to request a secure transport association to the destination.

The 'Resource-Priority' header field can be carried in the SIP message header or can be encapsulated in a message fragment carried in the SIP message body [RFC3420]. Encapsulation in S/MIME body parts allows the user to protect this header field against inspection or modification by proxies. However, in many cases, proxies will need to authenticate and authorize the request, so that encapsulation is undesirable.

Removal of a Resource-Priority header field or downgrading its priority value affords no additional opportunities to an adversary since that man-in-the-middle could simply drop or otherwise invalidate the SIP request and thus prevent call completion.

Only SIP elements within the same administrative trust domain employing a secure channel between their SIP elements will trust a Resource-Priority header field that is not appropriately signed. Others will need to authenticate the request independently. Thus, insertion of a Resource-Priority header field or upgrading the priority value has no further security implications except causing a request to fail (see discussion in the previous paragraph).

8.3 Anonymity

Some users may wish to remain anonymous to the request destination. Anonymity for requests with resource priority is no different than for any other authenticated SIP request. For the reasons noted earlier, users have to authenticate themselves towards the SIP elements carrying the request where they desire resource priority treatment. The authentication may be based on capabilities and noms, not necessarily their civil name. Clearly, they may remain anonymous towards the request destination, using the network-asserted identity and general privacy mechanisms [RFC3323][RFC3325].

8.4 Denial-of-Service Attacks

As noted, systems described here are likely to be subject to deliberate denial- of-service attacks during certain types of emergencies. DOS attacks may be launched on the network itself as well as its authentication and authorization mechanism. As noted, systems should minimize the amount of state, computation and network resources that an unauthorized user can command. The system must not amplify attacks by causing the transmission of more than one packet

to a network address whose reachability has not been verified.

9. IANA Considerations

9.1 IANA Registration of 'Resource-Priority' and 'Accept-Resource-Priority' Header Fields

[NOTE TO RFC EDITOR: Replace RFC XXXX with RFC number of this document.]

The following is the registration for the 'Resource-Priority' header field:

RFC number: XXXX
Header name: 'Resource-Priority'
Compact form: none

The following is the registration for the 'Accept-Resource-Priority' header field:

RFC number: XXXX
Header name: Accept-Resource-Priority
Compact form: none

9.2 IANA Registration for Option Tag resource-priority

RFC number: XXXX
Name of option tag: 'resource-priority'
Descriptive text: Indicates or requests support for the resource priority mechanism.

9.3 IANA Registration for Response Code 417

RFC number: XXXX
Response code: 417
Default reason phrase: Unknown Resource-Priority

9.4 IANA Namespace and Priority Registrations

Additional namespaces and priority values are registered with IANA. Within each namespace, the registration may indicate the relative precedence levels, expressed as an ordered list. New labels should not be added to existing namespaces. The registration MUST describe, in the registration itself or by reference, how SIP elements should treat requests from that namespace, e.g., whether preemption or only preferential queueing are allowed. A reference to a stable external document, e.g., by the International Telecommunication Union, other SDOs or national regulatory bodies, suffices. An expert review, by

an expert designated by the Transport Area Director or designate, is required.

Namespaces do not describe how they relate to other existing namespaces, as each namespace is independent of other registrations.

Below is a template for the registration of a new namespace:

Namespace: Designation of the namespace, according to the BNF 'namespace' in Section 3.
Description: Description of the use and application of this particular namespace.
Documentation: If applicable, reference to a document describing the namespace in more detail.
Organization: If applicable, organization defininig this namespace. (For example, an IETF standards-track RFC could also define a namespace, not just an external organization.)
Policy: Either if not defined normatively elsewhere or for informative purposes, this element describes how a SIP element handles requests containing priority values with this namespace. There are many possible behaviors that cannot be exhaustively anticipated. Three common behaviors are preemption, precedence and threshold-exemption. Preemption means that a request with greater priority can displace an existing request with lower priority that is already in progress. Precedence means that a higher-priority request assumes a position in the queue ahead of a lower-priority request, but any in-progress request is not affected by its arrival. In addition, systems with preemption MAY specify whether requests that cannot obtain resources immediately are queued or rejected immediately. Threshold-exemption allows higher-priority calls to access resources, such as circuits, that are unavailable to lower-priority calls, e.g., because they are held in reserve. If the namespace does not define a particular policy, the term 'implementation-defined' should be used.
Priority values (least to greatest): A list of priority values, ordered from least to highest priority.

9.5 Initial Namespace Registrations

9.5.1 Namespace dsn

Namespace: dsn
Description: United States Defense Switched Network. The values are adopted from RFC 791 [RFC0791], omitting the levels "critic-ecp", "network control" and "internetwork control", as these are inappropriate here.

Documentation: ANSI T1.619, Section B1
Organization: United States Department of Defense, Defense Information Systems Agency (DISA).
Policy: Preemption with rejection.
Priority values (least to greatest): "routine", "priority", "immediate", "flash", "flash-override"

9.5.2 Namespace q735

Namespace: q735
Description: ITU Q.735.3 describes multi-level precedence and preemption in SS7. The namespace "q735" supports interworking with Q.735.3 (or equivalent) GSTN (ISDN) entities; this allows, for example, carrying information between Q.735.3 entities without loss of information. One or both of the SIP endpoints might be PSTN gateways.
Documentation: Q.735.3 [Q.735.3]
Organization: ITU-T
Policy: Precedence.
Priority values (least to greatest): "4", "3", "2", "1", "0"

9.5.3 Namespace DRSN

Namespace: drsn
Description: United States Defense Red Switched Network.
Organization: United States Department of Defense, Defense Information Systems Agency (DISA).
Policy: Preemption with rejection.
Priority values (least to greatest): "routine", "priority", "immediate", "flash", "flash-override", "flash-override-override"

10. Acknowledgments

Ben Campbell, Janet Gunn, Paul Kyzivat, Rohan Mahy, Mike Pierce and Samir Srivastava provided helpful comments.

Normative References

[Q.735.3] International Telecommunications Union, "Stage 3 description for community of interest supplementary services using Signalling System No. 7: Multi-level precedence and preemption", Recommendation Q.735.3, March 1993.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[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.

[RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[RFC3420] Sparks, R., "Internet Media Type message/sipfrag", RFC 3420, November 2002.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

Informative References

[I-D.ietf-ieprep-framework]
Carlberg, K., Brown, I. and C. Beard, "Framework for Supporting IEPS in IP Telephony", draft-ietf-ieprep-framework-08 (work in progress), February 2004.

[I-D.ietf-sip-authid-body]
Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-02 (work in progress), July 2003.

[I-D.ietf-sipping-trait-authz]
Peterson, J., "Trait-based Authorization Requirements for the Session Initiation Protocol (SIP)", draft-ietf-sipping-trait-authz-00 (work in progress), February 2004.

[RFC3323] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November 2002.

[RFC3324] Watson, M., "Short Term Requirements for Network Asserted Identity", RFC 3324, November 2002.

[RFC3325] 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.

[RFC3487] Schulzrinne, H., "Requirements for Resource Priority Mechanisms for the Session Initiation Protocol (SIP)", RFC 3487, February 2003.

[RFC3665] 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.

Authors' Addresses

Henning Schulzrinne
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027
US

Phone: +1 212 939 7042
EMail: hgs@cs.columbia.edu
URI: http://www.cs.columbia.edu

James Polk
Cisco
2200 East President George Bush Turnpike
Richardson, TX 75082
US

EMail: jmpolk@cisco.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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

SIP WG
Internet-Draft
Expires: January 14, 2005

C. Jennings
Cisco Systems
July 16, 2004

SIP Conventions for Voicemail URIs
draft-jennings-sip-voicemail-uri-02

Status of this Memo

Copyright Notice

Abstract

At the previous meeting, the working group consensus was that Cullen could not move this draft forward as an individual submission. However, there was no consensus to adopt it as a WG item. This version of the draft carefully documents exactly what it is that Cullen may not move forward.

SIP systems are often used to initiate connections to voicemail or unified messaging systems. This document describes a convention for forming SIP Request URIs that request particular services from unified messaging systems.

This work is being discussed on the sip@ietf.org mailing list.

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 [1].

2. Introduction

Unified messaging systems (UM) have developed out of traditional voice mail systems. They can be used for storing and interacting with voice, video, faxes, email and instant messaging. Users often use SIP to initiate communications with them. When a SIP call is routed to a UM, there is a requirement for the UM to be able to figure out several bits of information from the call so that it can deliver the desired services. The UM needs to know what mailbox should be used for the context of this call and possible reasons about what type of service is desired. This includes knowing the type of media (voice or IM for example). Many voice mail systems provide different greetings depending whether the reason the call was sent to voicemail was that the user was busy or because the user did not answer. All of this information can be delivered in existing SIP signaling from the call control that retargets the call to the UM, but there are no standardized conventions for describing how the desired mailbox and service requested are expressed. It would be possible for every vendor to make this configurable so that any site can get it to work; however, this is not a very realistic view of achieving interoperability among call control, gateways, and unified messaging systems from different vendors. These requirements and more are described in the History Requirements [9]. This document describes a convention for describing this mailbox and service information in the SIP URI so that vendors and operators can build interoperable systems. It meets some but not all of the requirements in [9].

The work in the History Info [10] draft can be used in similar systems. It is more comprehensive and covers a much wider set of requirements. A key difference from this system is that history allows the UM to look at the history of the call and decided on what the best treatment is for the call. This work requires the call control system to know something about the history of the call and specifically ask the UM to invoke a particular service.

If there were no need to interoperate with TDM based voicemail systems or allow TDM systems to use VoIP unified messaging systems, this problem would be a little easier. The problem that is introduced in the VoIP to TDM case is as follows. The SIP system needs to tell a PSTN GW both the subscriber's mailbox identifier (which typically looks like a phone number) and the address of the

voicemail system in the TDM network (again a phone number).

One topic that causes some confusion in the requirements for this has to do with the fact that the related PSTN mechanism can carry two addresses. These correspond to the original target of the call and the most recent target to which it has been redirected. In general, the original target is used to find the voice mail box. The target that most recently redirected is not as useful for voicemail but is very useful for billing. It is often used to bill the most recent portion of the call leg. This work addresses only the requirements for UM system, and billing is completely out of scope. The History draft is much more extensive and covers more cases that might be useful for billing, but this work does not.

The question has been asked why the To header cannot be used to understand which mailbox to use. One of the problems with this is that the call control proxies cannot modify the To header, and the UAC often set it incorrectly because they do not have information about the subscribers in the domain they are trying to call. This happens because the routing of the call often translates the URI multiple times before it results in an identifier for the desired user that is valid in the namespace that the UM system understands.

Another set of requirements that this mechanism can deal with is the call coverage naming issues. The problem is when Bill calls the 800 number that sends him to the helpdesk, the proxy may first fork the call to Alice (who works at the help desk), and then if Alice does not answer in a few seconds fork the call on to Bob (who also works at the helpdesk). Both Alice and Bob would like to be informed that the call was to the help desk before they answer the call. If neither answers, the call may get sent to the help desk's voice mailbox, not Bob's or Alice's.

3. Mechanism (UAS and Proxy)

The mechanism works by encoding the information for the desired service in the SIP URI that is sent to the UM system. Two chunks of information are encoded, the first being the target mailbox to use and the second being the SIP error code that caused this retargeting and indicates the desired service. The target mailbox can be put in the user part of the URI and is also put in a target URI parameter while the reason is put in the URI cause parameter. For example, if the proxy wished to use Alice's mailbox because her phone was busy, the URI sent to the UM system could be something like:

sip:alice@um.example.com;target=alice;cause=486

3.1 Target

The target parameter indicates the mailbox to use. In many cases the user portion of the SIP URI could be set to the same value but it does not have to be. For example in the case of a voice mail system on the PSTN, the user portion will contain the phone number of the voice mail system while the target will contain the phone number of the subscriber's mailbox.

3.2 Cause

The URI cause parameter is used to indicates the service that the UAS receiving the message should perform. It corresponds to the SIP Status-Code that results in the desired service being requested. A mapping between some common services and reason codes are:

Service	Cause Parameter
Busy	486
No answer	408
Unconditional	302
Deflect	487
No Contacts/Failure of UA	410

3.3 Retrieving Messages

The UM system MAY use the fact that the From header is the same as the URI target as a hint that the user wishes to retrieve messages.

4. Interaction with Netann

This approach is designed to interact well with the netann mechanism. A netann parameter[8] can be used to indicate exactly which initial prompt to play.

5. Interaction with History

The History mechanism[10] provides considerably more information that is useful for a UM system. This work does not stop a UM system from taking advantage of the History information if it is present and using that to handle the call.

6. Limitations of Voicemail URI

This system requires the proxy that is requesting the service to

understand what are valid targets on the UM system. For practical purposes this means that the approach is unlikely to work in many cases where the proxy is not configured with information about the UM system or if the proxy is not in the same administrative domain.

This system requires the call control proxy to know what it wants the UM to do instead of giving the UM system the information about the call that allows the UM system to decide what to do. For example, if a call to the help desk got forwarded first to Alice, then to Bob, then finally to the helpdesk UM system, the UM system may want to leave a copy of the message in the primary help desk mail box and also leave a copy in Alice's mailbox since she was the primary person at the helpdesk. In addition the UM system might want to page Alice, Bob and their supervisor to let them know that no one is staffing the help desk. This system does not provide enough information to the UM system about what happened to the call to meet the needs of a scenario such as the one above.

This system only works when the service the call control wants applied is fairly simple. For example it does not allow the proxy to express information like "Do not offer to connect to the target's colleague because that address was already tried".

Some systems have expressed requirements for the UAC to understand when the call is re-targeted and get updated information about where it was targeted to as the call proceeds. This work does not address this requirement - History does, as does the option of just sending a 1xx class message with a Reason header[7].

The mechanism in this document does not address any billing issues associated with forwarded calls. This is a separate problem.

These limitations discussed in this section are addressed by the History[10] work.

7. Examples

7.1 Proxy Forwards No Answer to Voicemail

In this example, Alice calls Bob. Bob's proxy runs a timer and determines that Bob has not answered his phone, and the proxy forwards the call to Bob's voicemail. Alice's phone is at 192.168.0.1 while Bob's phone is at 192.168.0.2. The important things to note is the URI in message F4.

```
F1: INVITE 192.168.0.1 -> proxy.example.com

INVITE sip:15555551002@example.com;user=phone SIP/2.0
Via: SIP/2.0/TCP 192.168.0.1:5060;branch=z9hG4bK-74bf9
From: Alice <sip:5551001@example.com>;tag=9fxced76sl
To: sip:15555551002@example.com;user=phone
Call-ID: c3x842276298220188511
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:x123456x@192.168.0.1;transport=tcp>
Content-Type: application/sdp
Content-Length: *Body length goes here*

* SDP goes here*


F2: INVITE proxy.example.com -> 192.168.0.2

INVITE sip:line1@192.168.0.2 SIP/2.0
Via: SIP/2.0/TCP 192.168.1.4:5060;branch=z9hG4bK-ik80k7g-1
Via: SIP/2.0/TCP 192.168.0.1:5060;branch=z9hG4bK-74bf9
From: Alice <sip:5551001@example.com>;tag=9fxced76sl
To: sip:15555551002@example.com;user=phone
Call-ID: c3x842276298220188511
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:x123456x@192.168.0.1;transport=tcp>
Content-Type: application/sdp
Content-Length: *Body length goes here*

* SDP goes here*


F3: 486 192.168.0.2 -> proxy.example.com

SIP/2.0 486 Busy Here
Via: SIP/2.0/TCP 192.168.1.4:5060;branch=z9hG4bK-ik80k7g-1
Via: SIP/2.0/TCP 192.168.0.1:5060;branch=z9hG4bK-74bf9
From: Alice <sip:5551001@example.com>;tag=9fxced76sl
To: sip:15555551002@example.com;user=phone;tag=09xde23d80
Call-ID: c3x842276298220188511
CSeq: 1 INVITE
Contact: <sip:x654321x@192.168.0.2;transport=tcp>
Content-Length: 0
```

F4: INVITE proxy.example.com -> um.example.com

```
INVITE sip:bob@um.example.com;target=bob;cause=486 SIP/2.0
Via: SIP/2.0/TCP 192.168.1.4:5060;branch=z9hG4bK-ik80k7g-2
Via: SIP/2.0/TCP 192.168.0.1:5060;branch=z9hG4bK-74bf9
From: Alice <sip:5551001@example.com>;tag=9fxced76sl
To: sip:15555551002@example.com;user=phone
Call-ID: c3x842276298220188511
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:x123456x@192.168.0.1;transport=tcp>
Content-Type: application/sdp
Content-Length: *Body length goes here*

* SDP goes here*
```

7.2 Zero Configuration UM System

In this example, the UM system has no configuration information specific to any user. The proxy is configured to pass a URI that provides the prompt to play and an email address in the user portion of the URI to send the recorded message to.

The call flow is the same as in the previous example except that the URI in F4 changes to specify the user part as Bob's email address, and the netann URI play parameter specifies where the greeting to play can be fetched from.

F4: INVITE proxy.example.com -> um.example.com

```
INVITE
   sip:bob@um.example.com;target=mailto:bob@example.com;cause=486;
   play=http://www.example.com/bob/busy.way
   SIP/2.0
Via: SIP/2.0/TCP 192.168.1.4:5060;branch=z9hG4bK-ik80k7g-2
Via: SIP/2.0/TCP 192.168.0.1:5060;branch=z9hG4bK-74bf9
From: Alice <sip:5551001@example.com>;tag=9fxced76sl
To: sip:15555551002@example.com;user=phone
Call-ID: c3x842276298220188511
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:x123456x@192.168.0.1;transport=tcp>
Content-Type: application/sdp
Content-Length: *Body length goes here*

* SDP goes here*
```

In addition, if the proxy wished to indicate a VXML script that the UM should execute, it could add a parameter to the URI in the above message that looked like:

voicexml=http://www.example.com/bob/busy.vxml

7.3 TDM Voice Mail Connected via a Gateway

In this example, the voicemail system has a TDM interconnect to a gateway to the VoIP system. Bob's mailbox is +1 555 555-1002 while the address of the voicemail system on the TDM network is +1 555 555-2000.

The call flow is the same as in the previous example except for the URI in F4.

F4: INVITE proxy.example.com -> gw.example.com

```
INVITE sip:+1-555-555-2000@um.example.com;user=phone;\
       target=tel:+1-555-555-1002;cause=486
       SIP/2.0
Via: SIP/2.0/TCP 192.168.1.4:5060;branch=z9hG4bK-ik80k7g-2
Via: SIP/2.0/TCP 192.168.0.1:5060;branch=z9hG4bK-74bf9
From: Alice <sip:5551001@example.com>;tag=9fxced76sl
To: sip:15555551002@example.com;user=phone
Call-ID: c3x842276298220188511
CSeq: 1 INVITE
Max-Forwards: 70
Contact: <sip:x123456x@192.168.0.1;transport=tcp>
Content-Type: application/sdp
Content-Length: *Body length goes here*

* SDP goes here*
```

7.4 Call Coverage

In this example a user on the PSTN calls a 800 number. The GW sends this to the proxy which recognizes that the helpdesk is the target. Alice and Bob are staffing the help desk and are tried sequentially but neither answers, so the call is forwarded to the helpdesk's voice mail.

The key item in this flow is that the invite to Alice and Bob looks like

```
INVITE sip:bob@um.example.com;target=helpdesk;cause=302 SIP/2.0
```

8. Syntax

This document updates the BNF in Section 25 of RFC 3261 [3] to add the target-param to the uri-parameter as shown below.

```
uri-parameter     =  transport-param / user-param /
                     method-param / ttl-param / maddr-param /
                     lr-param / other-param /
                     target-param / cause-param

target-param      =  "target"  EQUAL pvalue

cause-param       =  "cause" EQUAL Status-Code
```

9. PSTN Mapping

The mapping to PSTN protocol is important both for gateways that connect the IP network to existing TDM equipment, such as PBX's and voicemail systems, and for gateways that connect the IP network to the PSTN network. Both ISDN and ISUP have signaling for this information that can be treated as roughly equivalent for the purposes here.

The user portion of the URI SHOULD be used as the address of the voicemail system on the PSTN, while the target SHOULD be mapped to the original redirecting party on the PSTN side.

If the gateway and Proxy are in the same Trust Domain (defined in RFC 3325 [5]) and the Spec(T) includes compliance with this document and the Spec(T) asserts that the Proxy will do screening (whatever that means), then the gateway MAY claim it is screened; otherwise it SHOULD NOT assert that the diversion information is screened.

This draft says nothing about what to put into the redirecting numbers, as that has billing implications outside the scope of this work. The requirements here will work fine if the redirecting number is not set on the PSTN side. It is not recommended that the GW map the target information into the redirecting party information, but doing so is not in violation of this document.

The following SHOULD be used as the mapping between reason parameters and ISUP/ISDN redirect reason codes:

ISUP or ISDN	PSTN Reason	URI Cause Parameter
0000	Unknown	300
0001	Call forwarding busy or called DTE busy	486
0010	Call forwarding no reply	408
1111	Call forwarding unconditional or systematic call redirection	302
1010	Call deflection or call forwarding by the called DTE	487
1001	Call forwarding DTE out of order	410

The redirection counters SHOULD be set to one unless additional information is available.

10. IANA Considerations

This document adds a new value to the IANA registration in the sub-registry at http://www.iana.org/assignments/sip-parameters as defined in [6].

Parameter Name	Predefined Values	Reference
target	No	RFC XXXX
cause	Yes	RFC XXXX

Note to RFC Editor - replace XXXX with the RFC number of this document.

11. Security Considerations

This draft inherently discusses transactions involving at least 3 parties. This makes the privacy issues somewhat more complex.

The new URI parameters defined in this draft are generally sent from a Proxy or call control system to a unified messaging (UM) system or gateway to the PSTN, and then to a voicemail system. This tells the UM what service the proxy wishes to have performed. Just as any message sent from the proxy to the UM needs to be integrity protected, these need to be integrity protected. This stops attackers from doing things like causing a voicemail meant for the CEO of the company to go to an attacker's mailbox. RFC 3261 provides TLS and IPSEC mechanisms suitable for protecting against this.

The signaling from the Proxy to the UM will reveal who is calling whom and possibly some information about the presence of a user based on whether a call got sent to voicemail instead of being answered. This information can be protected by encrypting the SIP traffic between the Proxy and UM. Again, RFC 3261 contains mechanisms for accomplishing this using TLS and IPSEC.

The S/MIME based mechanisms in RFC 3261 will generally not be applicable for protecting this information because they are meant for end to end issues and this is primarily a middle to end scenario. Ongoing work on middle to end [11] may allow S/MIME based schemes to be used for protecting this information. These schemes would allow the information to be hidden and integrity protected if there was another administrative domain between the Proxy and UM. The current scheme is based on hop by hop security and requires all hops between the Proxy and UM to be trusted, which is the case in many deployment scenarios.

11.1 Integrity Protection of Forwarding in SIP

Forwarding of a call in SIP brings up a very strange trust issue. Consider the normal case of when A calls B, and then the call gets forwarded by a network element in the domain of B to C, and then C answers the call. A called B but ended up talking to C. This may be hard to separate from a man in the middle attack.

There are two possible solutions for this. One is that B sends back information to A saying don't call me, call C and signs it as B. The problem with this is that it reveals the fact that B has forwarded to C and often B does not want to do this. For example, B may be a work phone that has been forwarded to a mobile or home phone. The user does not want to reveal their mobile or home phone number but, even more importantly, does not want to reveal that they are not in the office but are instead working from home.

The other possible solution for this is that A needs to trust B only to forward to a trusted identity. This requires a hop by hop transitive trust such that each hop will only send to a trusted next hop and each hop will only do things that the user at that hop desired. This solution is enforced in SIP using the SIPS URI and TLS based hop by hop security. It protects from an off axis attack but if one of the hops is not trustworthy, the call may be subverted to an attacker.

Any redirection of a call to an attacker's mailbox is a very serious issue. It is trivial for the attacker to make the mailbox seem very much like the real mailbox and forward the message to the real mailbox so that the fact that the messages have been intercepted or

even tampered with is not detected.

11.2 Privacy Related Issues on the Second Call Leg

When A calls B and gets redirected to C, occasionally people say there is a requirement for the call leg from B to C to be anonymous. This is not the PSTN: there is no call leg from B to C; instead there is a VoIP session between A and C. If A had put a To header containing B in the initial invite message, unless something special is done about it, C will see that To header. If the person who answers phone C says "I think you dialed the wrong number, who were you trying to reach?" A will probably specify B.

If A does not want C to see that the call was to B, A needs a special relationship with the Proxy that does the forwarding so that it will not reveal that information, and the call should go through an anonymizer service that provides session or user level privacy (as described in RFC 3323 [4]) service before going to C. It's not hard to figure out how to meet this requirement, but it is difficult to figure out why anyone would want this service.

If B wants to make sure that C does not see that the call was to B, it is easier but a bit weird. The usual argument is Bill wants to forward his phone to Monica but does not want Monica to find out his phone number. It is a little weird that Monica would want to accept all Bill's calls without knowing how to call Bill to complain. The only person Monica will be able to complain to is Hillary who tried to call Bill. Several popular web portals will send SMS alert message about things like stock prices and weather to mobile phone users today. Some of these contain no information about the account on the web portal that imitated them, making it nearly impossible for the mobile phone owner to stop them. This anonymous message forwarding has turned out to be a really bad idea even where no malice was intended. Clearly some people are fairly dubious about the need for this, but never mind: let's look at how it is solved.

In the general case, the proxy needs to route the call through an Anonymization Service and everything will be cleaned up. Any Anonymization service that performs the "Privacy: Header" Service in RFC 3323 [5] MUST remove the reason and target URI parameters from the URI. RFC 3325 already makes it pretty clear you would need to clean up this sort of information.

There is a specialized case of some interest where the mechanism in this document is being used in conjunction with RFC 3325 and the UM and the Proxy are both in the trust domain. It this limited case, the problem that B does not want reveal their address to C can be solved by ensuring that the target parameter URI should never be in a

message that is forwarded outside the trust domain. If it is passed to a PSTN device in the trust domain, the appropriate privacy flag needs to be set in the ISUP or ISDN signaling.

12. Changes from 00 Version

The reason information was moved from being a tag in the URI to using the Reason header.

13. Changes from 01 Version

The reason information was moved from using the Reason header to being a tag in the URI.

Seriously, I'm not joking - this was the consensus at the last meeting.

14. Acknowledgments

Mary Barnes, Dean Willis, and Steve Levy have spent significant time with me helping me understand the requirements and pros and cons of various approaches. I would like to thank them very much for this, and since this is an acknowledgments section I would also like to acknowledge Rohan Mahy's help with the various documents on this subject and his encouragement to work on a solution that brings some consensus to this topic.

15. References

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

[5] 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.

[6] Camarillo, G., "The Internet Assigned Number Authority (IANA) Universal Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", draft-ietf-sip-uri-parameter-reg-02 (work in progress), June 2004.

[7] Schulzrinne, H., Oran, D. and G. Camarillo, "The Reason Header Field for the Session Initiation Protocol (SIP)", RFC 3326, December 2002.

15.2 Informative References

[8] Burger, E., "Basic Network Media Services with SIP", draft-burger-sipping-netann-07 (work in progress), September 2003.

[9] Barnes, M., "SIP Generic Request History Capability Requirements", draft-ietf-sipping-req-history-04 (work in progress), June 2003.

[10] Barnes, M., "An Extension to the Session Initiation Protocol for Request History Information", draft-ietf-sip-history-info-01 (work in progress), October 2003.

[11] 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.

Author's Address

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 902-3341
EMail: fluffy@cisco.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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

SIP
Internet-Draft
Expires: January 16, 2005

C. Jennings
Cisco Systems
J. Peterson
NeuStar, Inc.
July 18, 2004

Certificate Management Service for SIP
draft-jennings-sipping-certs-04

Status of this Memo

Copyright Notice

Abstract

This draft defines a Credential Service that uses a SIP subscribe/ notify mechanism to discover other users' certificates and credentials and be notified about changes to these certificates. Other user agents that want to contact that AOR can retrieve these certificates from the server. The result is that widespread deployment of S/MIME in SIP is possible, because no extra expense or effort is required of the end user.

This work is being discussed on the sipping@ietf.org mailing list.

Table of Contents

1. Introduction

SIP provides a mechanism for end to end encryption and integrity using S/MIME, and several security properties of SIP depend on S/MIME. S/MIME has not been widely implemented or deployed due to the complexity of providing a reasonable key management infrastructure. This document proposes a way to address certificate discovery, retrieval, and management for SIP deployments. It follows the Sacred Framework RFC 3760 [7] for management of the credentials. Combined with the Identity [2] work, this work allows users to have certificates that are not signed by any well known certificate authority while still strongly binding the user's identity to the certificate. This mechanism allows UAs such as IP phones to enroll and get their credentials without any more configuration information than they commonly have today, without any extra effort or key clicks by the end user, and without any extra expense for the end user. This mechanism also lets the UA discover and retrieve the public certificate for any other user and find out about certificate revocations.

The general approach is to provide a new SIP service referred to as a Credential Server that allows UAs to subscribe to some other user's certificate. The certificate is delivered in a SIP NOTIFY to the UA that subscribes. The identity of the certificate can be vouched for using Identity [2] work. The Credential Service can manage public certificates as well as credentials that include the user's private key. The user can install new credentials to the Credential Server using a SIP PUBLISH. The Credential Server authenticates UAs that are changing credentials or requesting private keys using a shared secret that both the UA and the Server know. Typically this will be the same shared secret that is used in Register with the Registrar for the domain.

The mechanism described in this document works for both self signed certificates and certificates signed by a well known certificate authority; however, it is imagined that most UAs using this would only use self signed certificates and would use an Authentication Service as described in [2] to provide strong identity binding to the certificates.

Previous versions of this draft (00 to 02) used HTTP instead of SIP for communicating with the Credential Server. The key difference with using SIP is that a certificate can be revoked by sending a new NOTIFY; in the HTTP based scheme, the certificates were cached for a predefined period of time, typically one day, so that a revocation could only take effect after the cache expired. The earlier version also did not deal with the SACRED problem and allowed several devices with the same AOR to all have different private keys. This resulted

in very large SIP message and was looking fairly unwieldy; so now, the UAs for one AOR share private keying material and use the SACRED framework to move it between devices.

This basic approach of this work is independent of the details of body modification [13] and identity discussions. However, the choices made there will affect the mechanisms used to implement the approach described here.

2. 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 [5].

Certificate: An X.509 style certificate containing a public key and a list of identities in the SubjectAltName that are bound to this key. The certificates discussed in this draft are generally self signed and use the mechanisms in the Identity work [2] to vouch for their validity.

Credential: For this document, this means the combination of a certificate and the associated private key.

3. Goals

- Allow negotiation of E2E encrypted sessions.
- Allow end to end encryption and integrity of SIP bodies that may be delivered in SIP signaling, such as page mode MESSAGEs or NOTIFY bodies in presence.
- Work for users with multiple UA devices.
- Provide a certificate revocation mechanism.

4. UA Discovering Certificates

UAs discover certificates by sending a SUBSCRIBE with an event type of pkix-cert to the AOR for which a certificate is desired. This could be a SIP or tel URL. The resulting NOTIFY will contain an application/pkix-cert body which contains the certificates. The UA MUST follow the procedures in Section 11.1 to decide if the received certificate can be used. The UA needs to cache this certificate for future use. The certificate MUST be removed from the cache if it has expired, if it is updated by a subsequent NOTIFY, or if the subscription has been terminated. The NOTIFY containing a certificate must be signed by an Authentication Service as described in Identity. If the identity asserted by the Authentication Service does not match the identity requests, the certificates in the NOTIFY are discarded and MUST NOT be used.

5. UA Discovering and Publishing Credentials

UAs discover credentials by subscribing to their AOR with an event type of credential, which will result in a message containing both an application/pkix-cert body and an application/pkcs8 body that has the associated private key information for the certificate. The UA can change the user's certificate and private key by sending the server a PUBLISH[3] with an event type of credential that contains both an application/pkix-cert and an application/pkcs8 body.

The UA needs to authenticate to the Credential Server for these operations. The UA MUST use TLS to connect to the server. The UA may be configured with a specific name for the Credential Server; otherwise it defaults to the name of the domain in the User's AOR. The TLS connection MUST present a certificate that matches the expected name for the credential server, so that the UA knows it is talking to the correct server. If the certificate presented by the server does not match the expected server, the UA MUST terminate the connection and notify the user. If the UA does not do so, it may end up publishing its private key information to an attacker. The Credential Server will authenticate the UA using the usual SIP Digest mechanism, so the UA can expect to receive a SIP challenge to the SUBSCRIBE or PUBLISH messages.

The application/pkix-cert body is a DER encoded X.509 certificate [10]. The application/pkcs8 bodies contains a DER encoded PKCS #8 object that contains the private key. The PKCS #8 objects MUST be of type PrivateKeyInfo. The integrity and confidentiality of the PKCS #8 objects is provided by the TLS transport. The transport encoding of all the MIME bodies is binary.

6. Credential Server Behavior

The Credential Server stores credentials for users and can provide the credentials or certificates to other user agents. The credentials are indexed by an URI that corresponds to the AOR of the user. When a UA requests a public certificate with a SUBSCRIBE, the server sends it in a NOTIFY and sends a subsequent NOTIFY any time it changes. When a credential is requested, the Server digest challenges the requesting UA to authenticate it so that the Server can verify that the UA is authorized to receive the requested credentials.

When the Credential Server receives a SUBSCRIBE for a certificate, it first checks to see if it has credentials for the requested URI. If it does not it returns a response indicating the user was not found. Otherwise it sets up a subscription and forms a NOTIFY with the certificate in the body and the From header field value set to the

request URI of the SUBSCRIBE. It MUST send this NOTIFY through an Authentication Service (as described in Identity [2]) or implement an Authentication Service itself. The Server is encouraged to keep the subscriptions active for AORs that are communicating frequently but MAY unsubscribe at any point of time. Any time the credentials for this URI change, the Server MUST send a new NOTIFY to any active subscriptions.

When a Credential Server receives a SUBSCRIBE for a credential, the Server has to authenticate and authorize the UA and validate that adequate transport security is being used. The Server MUST digest challenge the UA to authenticate the UA and then decide if it is authorized to receive the credentials.

Once the UA has authenticated with the Server, the Server can set up a subscription and send a Notify message that MUST contain the credentials. This NOTIFY message is sent thought an Authorization Service in the same way as the certificate subscriptions. If the credential changes, the Server MUST terminate any current subscriptions and force the UA to re-authenticate. This is so that if a secret for retrieving the credentials gets compromised, the rogue UA will not continue to receive credentials after the compromised secret has been changed.

When the Credential Server receives a PUBLISH to update credentials, it MUST authenticate and authorize this the same way it does the subscriptions for credentials. If this succeeds, the Server updates the credential for this URI and processes all the active subscriptions to this URI as described above.

7. Negotiation of Secure Session

SIP uses an offer/answer negotiation mechanism[16] that describes sessions using SDP that may contain keying material, described in [14], for media protocols such as SRTP [15]. This keying material needs to be protected, and SIP does this by encrypting the SDP bodies using S/MIME.

If a UA receives both an unencrypted and an encrypted SDP offer in an multipart/alternative body, it interprets these as it would a normal multipart alternative as defined in RFC 2046 [17], which means it picks the last alternative that it can support. Any bodies that cannot be decrypted are treated as unsupportable. The sending UA should generally put encrypted offers after unencrypted ones, since encrypted ones are preferred. The UA constructs the answer to the offer as it normally would and may include both encrypted and unencrypted versions of the answer using multipart/alternative. The only wrinkle here is that if the UA sent multiple bodies with an

offer, it needs to be able to match the answer (or answeres) to the offer that was chosen.

The UA that made the offer can uniquely identify the various MIME bodies using a MIME Content-ID header. However, the UA sending the answers needs to provide the label of the Content-ID in the response. Solutions were considered that put the Content-ID identifier in a SIP Header, a MIME header, or an SDP attribute. Since the issue here is fundamentally about providing information that is all at the MIME level about the relation between one set of multipart/alternatives and the other MIME body that is being sent, the best solution seems to involve passing this tag at the MIME level. A new MIME header called "Content-Related-To" updates RFC 2045 with:

```
rid := "Content-Related-To" ":" msg-id
```

and adds "[rid CRLF]" to the entity-headers.

The identifier supplied in the Content-Related-To header must be a valid Content-ID from a previous MIME message that this body is related to.

The UA looks at the multipart/alternatives and selects the best one it can use. It MUST include a Content-Related-To in the MIME for the answer that copies the tag from the related Content-ID header of the offer body it has chosen to use.

In a typical call from Alice to Bob, Alice would first subscribe to Bob's certificate. If this worked, then Alice would send an Invite to Bob that contained an RTP session in unencrypted SDP and an SRTP session in encrypted SDP. Bob would select the SRTP session and send an answer with encrypted SDP selecting the SRTP session. Both Alice's and Bob's UAs would indicate to the user that a secure call had been negotiated. Alice and Bob could note that the call was secure and adjust their conversation accordingly.

8. Encrypting Bodies of SIP messages

Applications such as presence and 911 location information result in information with significant privacy requirements being sent in SIP. Particular MIME types may define special meanings when both an encrypted and unencrypted body are received, but, unless otherwise specified, the UA SHOULD use the encrypted version if it can decrypt it, and ignore the unencrypted version. There is no requirement for the two versions to have the same information. For example, a page mode message could have an unencrypted version that said "I'm in the Middle East visiting people" while the encrypted version had much

more sensitive information like "I'm over at Osama's house at 21.25'24"N 39.49'24"E". Depending whether the receiving device can decrypt this or not, a different message gets displayed to the receiving user.

9. Signing Bodies of SIP message

In general, signing messages with self-signed certificates is not that useful unless some other means is used to vouch that the certificate has some meaning. If the Authentication Service is used to do this, then the Authentication Service is providing integrity across all the bodies and binding them with an identity. In this case, the additional signature becomes redundant. Because of this, it is recommended that signing bodies SHOULD NOT be used if the certificate is a self signed certificate.

10. Examples

In all these examples, large parts of the message are omitted to highlight what is relevant to this draft. The lines in the examples that are prefixed by $ represent encrypted blocks of data.

10.1 Encrypted Page Mode IM Message

In this example, Alice sends Bob an encrypted page mode instant message. If Alice does not already have Bob's public key from previous communications, she fetches Bob's public key from Bob's credential server:

```
SUBSCRIBE sip:bob@biloxi.example.com SIP/2.0
...
Event: certificate
```

The credential server responds with the certificate in a NOTIFY.

```
NOTIFY alice@atlanta.example.com  SIP/2.0
Subscription-State: active; expires=7200
....
From: <sip:bob@biloxi.example.com>;tag=1234
Identity: "12dsfsdk2389403823cbed"
Identity-Info: sips:billoxi.example.com
....
Event: certificate
Content-Type: application/pkix-cert

< certificate data >
```

Next Alice sends a SIP MESSAGE message to Bob:

```
MESSAGE sip:bob@biloxi.example.com SIP/2.0
...
Content-Type: application/pkcs7-mime

$ Content-Type: text/plain
$
$ < encrypted version of "Hello" >
```

10.2 SRTP Phone Call

In this example, Alice calls Bob and offers both an RTP and an SRTP session. The SDP for the SRTP session contains the SRTP keying material and is encrypted with S/MIME. If Alice does not already have Bob's public key from previous communications, she fetches Bob's public key from Bob's credential server in the same way as shown in the previous example.

Alice sends an INVITE to Bob that offers two alternative SDP bodies, one of which is encrypted and contains the SRTP keying information. The

```
INVITE sip:bob@biloxi.example.com SIP/2.0
...
Content-Type: multipart/alternative;boundary=boundary

--boundary
Content-ID: 123
Content-Type: application/sdp
Content-Disposition: session

< SDP offer for ordinary RTP only >
--boundary
Content-ID: 456
Content-Type: application/pkcs7-mime
Content-Disposition: session

$ Content-Type: application/sdp
$
$ < encrypted SDP with key for SRTP >
--boundary
```

If Bob's UA does not have Alice's public key, Bob's UA would fetch it

as shown in the previous example. Assuming that Bob's UA supported encryption, it would select the second alternative offer and construct an appropriate answer. The 200 includes the MIME Content-Related-To header that indicates which alternative MIME body was chosen.

```
200 OK
...
Content-ID: 789
Content-Related-To: 456
Content-Type: application/pkcs7-mime
Content-Disposition: session

$ Content-Type: application/sdp
$
$ < encrypted SDP with key for SRTP >
```

10.3 Setting and Retrieving UA Credentials

When Alice's UA wishes to publish Alice's public and private keys to the Credential Server, it sends a PUBLISH message like the one below. This must be sent over a TLS connection in which the other end of the connection presents a certificate that matches the Credential Server for Alice and digest challenges the message to authenticate her.

```
PUBLISH sip:alice@atlanta.example.com SIP/2.0
...
Content-Type: multipart/mixed;boundary=boundary

--boundary
Content-ID: 123
Content-Type: application/pkix-cert
Content-Disposition: session

< Public certificate for Alice >
--boundary
Content-ID: 456
Content-Type: application/pkcs8
Content-Disposition: session

< Private Key for Alice >
--boundary
```

If one of Alice's UAs subscribes to the credential event, the UA will be digest challenged, and the NOTIFY will include a body similar to

the one in the PUBLISH section above.

11. Security Considerations

This whole scheme is highly dependent on trusting the operators of the Credential Server and trusting that the Credential Server will not be compromised. The security of all the users will be completely compromised if the Credential Server is compromised.

This work requires the TLS session to be used for communications to the Credential Server. Failing to use TLS or selecting a poor cipher suite (such as NULL encryption) will result in credentials being sent unencrypted over the network and will render the whole system useless. Implementation really must use TLS or there is no point in implementing any of this. In addition, the correct checking of chained certificates as specified in TLS [11] is critical for the client to authenticate the server.

If a particular credential needs to be revoked, the new credential is simply published to the Credential Server. Every device keeping this current in its cache will have a subscription to the credential and will rapidly (order of seconds) be notified and replace its cache. Clients that are not subscribed will subscribe and get the new certificate, so they will not end up using the old invalid certificate.

11.1 Trusting the Identity of a Certificate

When a UA wishes to discover the certificate for sip:alice@example.com, the UA subscribes to the certificate for alice@example.com and receives a certificate in the body of a SIP Notify message. The term original URI is used to describe the original URI that was subscribed to.

If the certificate is signed by a trusted CA, and one of the names in the SubjectAltName matches the original URI, then this certificate MAY be used but only for exactly the Original URI and not for other identities found in the SubjectAltName. Otherwise, there are several steps the UA MUST perform before using this certificate.

- The From header in the NOTIFY message MUST match the original URI.
- The UA MUST check the Identity header as described in the Identity [2] work to validate that bodies have not been tampered with and that an Authentication Service has validated this From header.
- The UA MUST check the validity time of the certificate and stop using the certificate once it is invalid.
- The certificate MAY have several names in the SubjectAltName but the UA MUST only use this certificate when it needs the certificate for the identity in the Original URI. This means that

the certificate should only be indexed in the certificate cache by the value of the original URI, not by the value of all the identities found in the SubjectAltName.

These steps result in a chain of bindings that result in a trusted binding between the original URI and a public key. The Original URI is forced to match the From. The Authentication Service validates that this message did come from the identity claimed in the From and that the bodies and From have not been tampered with. The certificate in the body contains the public key for the identity. Only the UA that can authenticate as this user can tamper with this body, so the owner of the identity can provide a false public key but other users cannot. This chain of assertion from original URI, to From, to body, to public key is critical to the security of the mechanism described in this document. If any of the steps above are not followed, this chain of security will be broken and the system will not work.

11.2 Conformity to the SACRED Framework

This work uses the security design outlined in the SACRED Framework [7]. Specifically, it follows the cTLS architecture described in section 4.2.2 of RFC 3760. The client authenticates the server using the server's TLS certificate. The server authenticates the client using a SIP digest transaction inside the TLS session. The TLS sessions form a strong session key that is used to protect the credentials being exchanged.

Credential Servers SHOULD implement the server name indication extensions in RFC 3546 [8] and they MUST support a TLS profile of TLS_RSA_WITH_AES_128_CBC_SHA as described in RFC 3268 [9] and a profile of TLS_RSA_WITH_3DES_CBC_SHA.

12. IANA

The MIME Content-Related-To header does not require any IANA actions.

12.1 Certificate Event Package

To: ietf-sip-events@iana.org
Subject: Registration of new SIP event package

Package Name: certificate

Is this registration for a Template Package: No

Published Specification(s): draft-jennings-sipping-certs

Person & email address to contact for further information:
Cullen Jennings <fluffy@cisco.com>

12.2 Credential Event Package

To: ietf-sip-events@iana.org
Subject: Registration of new SIP event package

Package Name: credential

Is this registration for a Template Package: No

Published Specification(s): draft-jennings-sipping-certs

Person & email address to contact for further information:
Cullen Jennings <fluffy@cisco.com>

12.3 PKCS #8

To: ietf-types@iana.org
Subject: Registration of MIME media type application/pkcs8

MIME media type name: application

MIME subtype name: pkcs8

Required parameters: None

Optional parameters: None

Encoding considerations: will be binary for 8-bit transports

Security considerations: Carries a cryptographic private key

Interoperability considerations: None

Published specification: draft-jennings-sipping-certs

Applications which use this media type: Any MIME-complaint transport

Additional information:
Magic number(s): None
File extension(s): .p8
Macintosh File Type Code(s): none

Person & email address to contact for further information:
Cullen Jennings <fluffy@cisco.com>

Intended usage: COMMON

Author/Change controller:
Cullen Jennings <fluffy@cisco.com>

13. References

13.1 Normative References

[1] RSA Laboratories, "Private-Key Information Syntax Standard, Version 1.2", PKCS 8, November 1993.

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

[3] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", draft-ietf-sip-publish-04 (work in

progress), May 2004.

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

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

[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] Gustafson, D., Just, M. and M. Nystrom, "Securely Available Credentials (SACRED) - Credential Server Framework", RFC 3760, April 2004.

[8] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[9] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

[10] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.

[11] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

13.2 Informational References

[12] Gutmann, P., "Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP", draft-ietf-pkix-certstore-http-07 (work in progress), May 2004.

[13] Mahy, R., "Pros and Cons of allowing SIP Intermediaries to add MIME bodies", draft-mahy-sipping-body-add-00 (work in progress), July 2004.

[14] 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.

[15] Baugher, M., McGrew, D., Naslund, M., Carrara, E. and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

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

[17] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

Authors' Addresses

Cullen Jennings
Cisco Systems
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 902-3341
EMail: fluffy@cisco.com

Jon Peterson
NeuStar, Inc.
1800 Sutter St
Suite 570
Concord, CA 94520
US

Phone: +1 925/363-8720
EMail: jon.peterson@neustar.biz
URI: http://www.neustar.biz/

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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

SIPPING WG
Internet-Draft
Expires: August 7, 2004

C. Jennings
Cisco Systems, Inc.

February 7, 2004

Updating the Connected Party in SIP
draft-jennings-sipping-connected-00.txt

Status of this Memo

Copyright Notice

Abstract

Communication systems often have situations in which the party at the far end of the call changes, making it necessary to have a way to notify the near end of the new identity at the far end. This draft discusses ways to update this "connected party" information in SIP.

This work is being discussed on the sipping@ietf.org mailing list.

Table of Contents

1. Conventions and Definitions

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

2. Introduction

SIP[5] initiates sessions but it also provides information on the identities of the parties at both ends of a session. This is necessary as users find it very desirable for knowing how to deal with the communications that SIP is initiating. As a call proceeds, these identities may change. This can happen for many reasons: calls are forwarded, calls are parked and picked up, calls are transferred, calls are queued to be picked up by a pool of agents, and so on. The use cases here are split into two categories: where the identity change happens in an early dialog or a connected call, or where the callee's or the caller's identity changes. How the identity of the called party is provided to the caller is not well defined in SIP.

This is early work discussing the problems and looking at possible solutions to this problem. It needs considerable work and consideration for how the approach would interact with existing systems. It is being discussed on the sipping@ietf.org mailing list.

3. Use Cases

Identity information often contains both a human readable name and a network address such as a phone number or SIP URL.

3.1 Connected UAS or UAC Updated

The classic case is when Alice calls Bob who forwarded to Charlie who answers the phone. Alice would like to know she is talking to Charlie not Bob.

These cases do not involve early dialogs. Since the call has been established, the use cases are symmetrical. Either the UAS or UAC could need to be updated in any of these cases.

The classic case is when a call is transferred.

The UAS has originally answered the call with the identity Help Desk but wishes to change it, mid-call, to Alice.

3.2 Early UAC Updated

In a typical example, phone A rings but user B remotely picks up the

ringing phone from a different one.

Call queuing in an ACD is another example. The call may initially be queued for the "Help Desk" but later be moved to ring Bob's phone.

It should be noted that the history requirements[9] cover this case.

3.3 Early UAS Updated

First, Alan dials Charlie on behalf of another user, Bob in this case. While Charlie's phone is ringing, Alan steps out of the call and lets Bob talk to Charlie.

4. Solutions Using Transfer Mechanisms

Many of these use cases seem to come up because there is a B2BUA involved. This B2BUA may be acting as a call control system or it may be a PSTN inter-working GW which is effectively a B2BUA with the PSTN connecting the two UAs that form the B2BUA. It's fine to have a B2BUA connected to a P2P system; however, the whole design of the B2BUA concept in SIP is to make B2BUA just look like a valid peer to the P2P system.

These solutions are derived by looking at what would happen if approximately the same functionality were done in a P2P system, and looking at signaling from the point of view of the UA that is receiving the updated identity.

4.1 Connected UAS or UAC Updated

When A is in a connected call with B and B wishes to update the identity that A sees to C, this is the same from A's point of view as B transferring the call to C. In this case the identity can be updated by sending a new INVITE to A with the From set to C and including a replaces header to indicate it replaces the original call.

This approach may have bad interaction with who gets billed.

4.2 Early UAC Updated

Here the UAS wishes to change its identity and update the UAC. The UAS sends back a new early dialog with the new identity. From the UAC point of view, this looks like the call to a UAS that forked to form a new early dialog with a different UAS. This might have bad interactions with early media.

Trying to solve this the same way as the when the UAC is connected

does not work because the new INVITE with the replaces results in a completely wrong view about which UA is in the ringing state and which is already off hook.

4.3 Early UAS Updated

Here the UAC wishes to change its identity and update it on all the UASs that it has forked to while it had an outstanding INVITE. The transfer issues draft [3] clearly points out that the UAS is a moving target due to forking with the "Consultative Turned Blind" issues. Changing the identity of UAC complicates this even further because the routing of the invite may have been dependent on the identity of the UAS. For example, calls from Private may go straight to voice mail while calls from Boss get forked to office phone and cell phone.

It is probable that whatever solution is chosen to deal with the "Consultative Turned Blind" problem in [3] can also be used to update identity in this scenario. The recommendation here is to hold off on defining something special for identity updates until the solution for the transfer issues problem is resolved.

The idea of sending a new INVITE with replaces, waiting a little while, and then sending a CANCEL to the original INVITE has been suggested. This fails in the case where the call is to the PSTN and forks to a different gateway.

5. Solutions Based on UPDATE

User agents could send UPDATE requests that do not contain SDP, but do change the To and From headers. The tags in the To or From headers would not be changed. Any UPDATES received by an RFC 2543 [10] system would fail to match the transaction due to the changed To or From. This would not result in the call failing but would only result in the identity failing to be displayed correctly.

Changing the To and From headers was contemplated in Section 12.2.1.1 of RFC 3261 which says "Usage of the URI from the To and From fields in the original request within subsequent requests is done for backwards compatibility with RFC 2543, which used the URI for dialog identification. In this specification, only the tags are used for dialog identification. It is expected that mandatory reflection of the original To and From URI in mid-dialog requests will be deprecated in a subsequent revision of this specification."

As an alternative to changing the To and From headers, a new Name header could be created that represents the identity of the sender of the message.

The biggest problem with these approaches is that they make it very hard to apply new policy to the call if the user's identity changes. Consider the example above. The identity of the caller determines where the call is forked; with this approach, the system would not change where the call was being forked if the identity of the caller changed.

Another issue is how this interacts with outstanding UPDATES that are being used for offer/answer negotiation.

6. Why PAI is Useless

PAI, described in RFC 3325 [4], was designed to work in very limited trust domains. It is primarily useful when a UAC wishes its identity to be anonymous to the UAS but the intermediate trust domain needs to know the UAC's identity for legal call trace reasons. There are no legal call trace requirements of this type from the UAS to the UAC. If the UAS wishes to change its identity to Anonymous, there is no need to carry the real identity along. It is important to understand this: the requirement solved by PAI is asymmetrical and does not apply in the backwards direction.

The requirements driving the work in this document are about informing the party at the far end of the call of a changed identity. PAI will never work for this. The fundamental requirement for PAI is that as the PAI crosses from one trust domain A to the next trust domain B, the next trust domain B must discard it unless it can independently verify it. This is so because if B passes it to any other element even inside its trust domain, those elements will assume the PAI is true even though B does not know this to be true. The end user device that is capable of displaying the PAI information is never in the same trust domain as the network elements, so it must discard this information as it comes in. RFC 3325 specifically says "However, if a User Agent Server receives a message from a previous element that it does not trust, it MUST NOT use the P-Asserted-Identity header field in any way." A UAS that sits in the user's hand is not going to be trusted by the "network" trust domain. Trust domains are symmetrical - you cannot have a trust domain in which the phone trusts the network but the network does not trust the phone.

To summarize, given two trust domains A and B, even if you passed the PAI around in responses in A, as soon as it got passed to B, it would be discarded or unusable. If it got passed in a response to a proxy in B, 3325 says that "the proxy MUST authenticate the originator of the message" which of course the proxy cannot do for a response, so it would have to drop the PAI. There is no use case in which the UA producing the PAI and the UA that could display a PAI to an end user

are in the same trust domain. A new header, like Name, could be defined as described in Section 5 but PAI is not useful for this.

7. Recommendations

Deprecating forking and early media do not seem feasible. The early unattended transfer problem has been floating around for a long time with no good solution. Using UPDATE or some new method to update connected party information after a dialog is formed seems very appealing. This name could be in a changed To or From header or in a new header. This same approach could be used when the caller identity changed in an early dialog. History seem like a good approach for coping with the ever changing identity of various UASs during early dialogs.

8. Acknowledgments

Thanks to Randy Baird and Rohan Mahy.

Normative References

[1] Sparks, R., "The SIP Referred-By Mechanism", draft-ietf-sip-referredby-03 (work in progress), August 2003.

[2] Biggs, B., Dean, R. and R. Mahy, "The Session Inititation Protocol (SIP) 'Replaces' Header", draft-ietf-sip-replaces-04 (work in progress), August 2003.

[3] Petrie, D., "SIP Transfer Issues", draft-petrie-sipping-xfer-issues-00 (work in progress), October 2003.

[4] 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.

[5] 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.

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

Informative References

[7] Elwell, J., "Interworking between SIP and QSIG", draft-ietf-sipping-qsig2sip-02 (work in progress), August 2003.

[8] Venkataramanan, V., "Enhancements to Asserted Identity to Enable Called Party Name Delivery using SIP", draft-venkatar-sipping-called-name-00 (work in progress), June 2003.

[9] Barnes, M., "SIP Generic Request History Capability Ãû Requirements", draft-ietf-sipping-req-history-04 (work in progress), June 2003.

[10] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

Author's Address

Cullen Jennings
Cisco Systems, Inc.
170 West Tasman Dr.
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 902 3341
EMail: fluffy@cisco.com

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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

SIP
Internet-Draft
Expires: January 10, 2005

H. Tschofenig
Siemens
J. Peterson
NeuStar, Inc.
J. Polk
Cisco
D. Sicker
CU Boulder
M. Tegnander
Siemens
July 12, 2004

Using SAML for SIP
draft-tschofenig-sip-saml-00

Status of this Memo

Copyright Notice

Abstract

This document describes how to use the Security Assertion Markup Language (SAML) to offer trait-based authorization. As such, it

provides an alternative to existing authorization mechanisms for SIP.

Table of Contents

1. Introduction

This document proposes a method for using the Security Assertion Markup Language (SAML) in collaboration with SIP to acommodate richer authorization mechanisms and enable trait- based authorization where you are authenticated using roles or traits instead of identity. A motivation for trait based authorization and some scenarios are presented in [I-D.ietf-sipping-trait-authz].

Security Assertion Markup Language (SAML)[I-D.saml-tech-overview-1.1-03] is an XML extension for security information exchange. It is being developed by OASIS. SAML enables users to gain access to multiple website resources without having to re-authenticate every time the domain changes. The first authentication would be transferred to subsequent domains using SAML.

To provide trait-based authorization a few solutions are possible: authorization certificates, SPKI or extensions to the authenticated identity body [I-D.ietf-sip-authid-body]. The authors selected SAML due to the amount of work done in the area of SAML which provides some assurance that this technology is mature enough.

2. Terminology

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 [RFC2119].

The SIP entity 'Authentication Service' was introduced with [I-D.ietf-sip-identity].

The Authentication Service is the entity that authenticates and authorizes a user and creates an Assertion. This entity is the equivalent of the asserting party in the SAML terminology.

For terminology related to SAML the reader is referred to [I-D.saml-tech-overview-1.1-03].

3. Goals and Non-Goals

This document tries to accomplish the following goals:

- This document defines how SAML assertions are carried in the SIP header. As such, the usage of SAML Assertions within SIP can be seen as a SAML profile.

- The requirements and scenarios defined in [I-D.ietf-sipping-trait-authz] are compared to the solution described in this document by utilizing SAML assertions.

The following issues are outside the scope of this document:

- The configuration of the Authentication Service in order to attach certain assertions is outside the scope of this specification and might depend on the environment where SIP is used. To avoid restricting the functionality of SIP either as an in-band or an out-of-band mechanism, it can be defined to trigger the inclusion of SAML assertions. XCAP[I-D.draft-ietf-simple-xcap-02] is a possible mechanism to configure the behavior of the Authentication Service in an out-of-band fashion.

- The attributes stored in Assertions are, for example, roles, membership to a certain organization, specific access rights or information about the authentication. A definition of most of these attributes is application dependent and not defined in this document. Since the attributes need to be understood by the entity verifying the assertions it might be necessary to describe common attributes in a future version of this document. Note that some attribute definitions are already available with SAML (such as attributes providing information about the authentication procedure).

- SIP is not used as a request/response protocol for obtaining Assertions (although possible). Such a protocol is, for example, required between the Relying Party and the Asserting Party to fetch an Assertion based on a received Artifact. Note, however, that SIP is still implicitly used to request Assertions.

4. SAML Introduction

In SAML there are three main entities: the user, the asserting party and the relying party. The asserting party is asserting that a particular user has been given proper authorization. The relying party has to trust the asserting party with regard to the provided information and then decides whether or not to accept the assertions provided, giving different levels of privileges.

The components of SAML are:

- Assertions/Artifact
- Request/Response protocols
- Bindings
- Profiles

We describe each in turn below

4.1 Assertions

An Assertion is a package of information including authentication statements, attribute statements and authorization decision statements. All kinds of statements do not have to be present, but at least one. An Assertion contains several elements:

Issuing information:

Who issued the assertion, when was it issued and the assertion identifier.

Subject information:

The name of the subject, the security domain and optional subject information, like public key.

Conditions under which the assertion is valid:

special kind of conditions like assertion validity period, audience restriction and target restriction.

Additional advice:

explaining how the assertion was made, for example.

In an authentication statement, an issuing authority asserts that a certain subject was authenticated by certain means at a certain time.

In an attribute statement, an issuing authority asserts that a certain subject is associated with certain attributes which has certain values. For example, user jon@cs.example.com is associated with the attribute 'Department', which has the value 'Computer Science'.

In an authorization decision statement, a certain subject with a certain access type to a certain resource has given certain evidence that the identity is correct. Based on this, the relying party then makes the decision on giving access or not. The subject could be a human or a program, the resource could be a webpage or a web service, for example.

4.2 Artifact

The artifact used in the Browser/Artifact profile, is a base-64 encoded string which is 40 bytes long. 20 bytes consists of the typecode, which is the source id. The remaining 20 bytes consists of a 20-byte random number that servers use to look up an assertion. The source server stores the assertion temporarily. The destination server receives the assertion and pulls the data from the artifact on the source site. The purpose of the artifact is to act as a token who references an assertion for the subject who holds the artifact.

4.3 Request/Response protocol

SAML defines a request/response protocol for obtaining Assertions. The request asks for an Assertion or makes queries for authentication, attribute and authorization decisions. The response is carrying back the requested Assertion. The XML format for protocol messages are defined within an XML schema in TBD.

4.4 Bindings

The bindings in SAML maps between the SAML protocol and a transport and messaging protocol. With SAML Version 1.1 there is only one binding specified, which is SAML embedded in SOAP-over-HTTP. In a binding, a transport and messaging protocol is used only for transporting the request/response mechanism.

4.5 Profiles

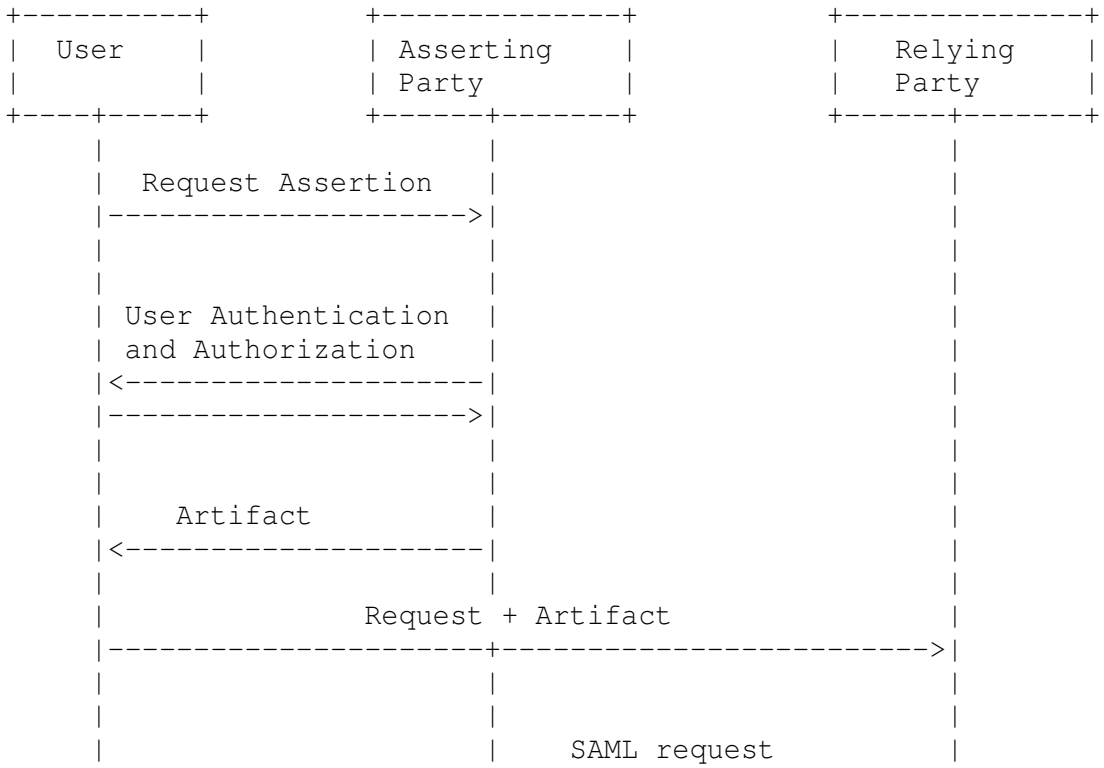
When using a profile, SAML is used to provide assertions about a resource in the body of the message itself. In Version 1.1 of SAML, there are two profiles specified, the Browser/Artifact profile and the Browser/POST profile. The Browser/Artifact profile respresents a "pull" model, where a special reference to the assertion called an artifact, is sent to the relying party from the asserting party. The artifact is then used to "pull" the assertion from the asserting party. The Browser/POST profile represents a "push" model, where an assertion is posted (using the HTTP POST command) directly to the relying party. These two models are described in Figure 1 and in Figure 2.

5. Assertion Handling Models

As mentioned in Section 4.5, two main models can be used in SAML and therefore also with the SIP-SAML extension defined in this document: The Push and the Pull model.

In the Pull model the end host requests an assertion from the Asserting Party and receives, after successful authentication and authorization, an Artifact. The Artifact is a special form of an Assertion. This Artifact can be compared with the call-by reference approach where a reference to the Assertion is stored at the Asserting Party and can later be referenced. The Relying Party later fetches the SAML Assertion after receiving a request by the user which includes the Artifact. For communicating the SAML request and response messages, a separate message exchange is needed with a protocol such as SOAP or HTTP. That is outside the scope of this document.

Note that this exchange also allows the Artifact to be bound to a particular signaling session by attaching the assertion to the service request. This requires the Asserting Party to participate in the signaling message exchange and provides stronger security properties but removes the property of "single sign-on".



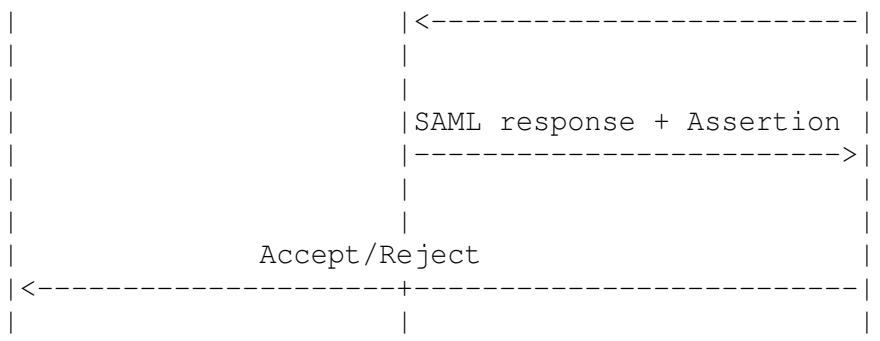


Figure 1: SAML Pull model

With the Push model, the user requests an Assertion from the Asserting Party. The user can also trigger the Asserting Party to attach an Assertion to the request. The Assertion, which is added to the service request, can be verified by the Relying Party without additional protocol interactions with the Asserting Party. The Assertion therefore contains enough information to authorize the service request. Using programming languages, a call-by value is implemented.

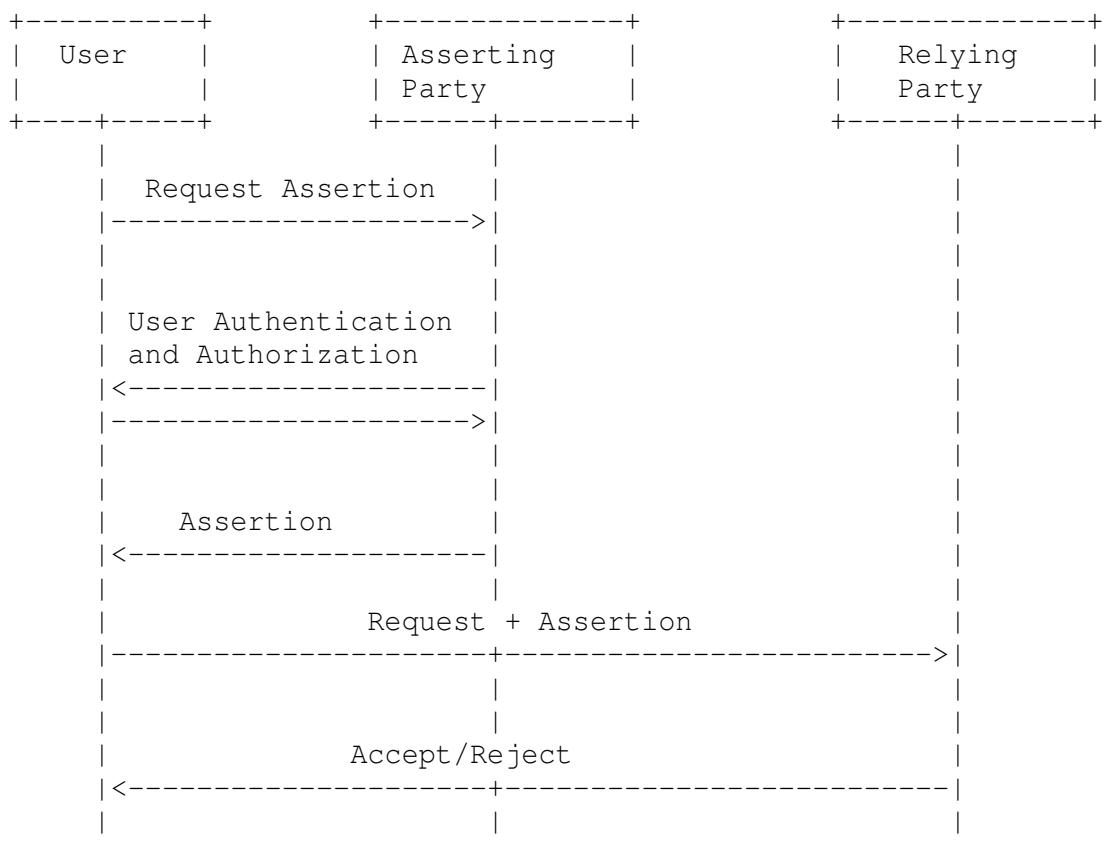


Figure 2: SAML Push model

The usage of SAML in HTTP-based environments and in SIP is, however, affected by some architectural differences. The main goal of realizing single-sign-on (SSO) functionality in HTTP is not the goal of this extension to SIP.

The function of the entities in the Push and the Pull model are shown in Figure 1 and in Figure 2.

The user has to request an Assertion at the Asserting Party and both entities mutually authenticate each other. The requested Assertion is sent to the user in a confidential manner to prevent eavesdroppers to obtain this Assertion. The Relying Party trusts the Asserting Party. It is assumed that the accessed resource is located at the Relying Party and that this entity does not become malicious or that it does not act on behalf of the user to impersonate him or her to other parties with regard to access to this resource. To prevent some degree of misuse, attributes in the Assertion limit its applicability for certain applications, servers or time frame.

Signaling in SIP can, however, involve a number of entities in more

complex scenarios. As an example, the scenario addressed in [I-D.ietf-sip-identity] aims to replace end-to-end authentication via S/MIME by a "mediated authentication architecture". The end hosts only need to be able to verify Assertions signed by an Authentication Service which guarantees that the sender was authenticated.

Directly applying SAML to such a scenario, however, causes a problem: a SIP proxy, which securely receives a SAML Assertion (such as confidentially protected to prevent eavesdroppers between the SIP UA and the SIP proxy to learn the Assertion), can store this Assertion to impersonate the user in future requests towards other SIP end users. The fact that multiple parties see the Assertion along the path (i.e., SIP proxies) make the situation worse. The Assertion might include some attributes which restrict its usage (such as lifetime or indication of a particular resource) but they cannot fully prevent impersonation. If intermediate SIP proxies should also be able to process the Assertion then it cannot be bound to a particular receiver - the intermediate SIP proxies might not even be known to the SIP end host.

This problem appears if SAML Assertions are not bound to a particular protocol run. Binding the Assertion to a particular session is not useful in the context of SSO but in many SIP scenarios there seems to be a problem if such a binding is not provided. There is little usage in requesting Assertions from a separate Authentication Service for every SIP message exchange since the additional latency and performance impact could potentially be large.

6. Scenarios

This section shows message flows based on scenarios in [I-D.ietf-sipping-trait-authz] enriched with a SAML based solution. Section 6.1 provides an example of enhanced network asserted identities and Section 6.2 shows a SIP conferencing scenario with role-based access control using SAML. A future version of this document will cover more scenarios from [I-D.ietf-sipping-trait-authz].

6.1 Network Asserted Identities

Figure 3 shows an enhanced network asserted identity scenario based on [I-D.ietf-sip-identity] which again utilizes extensions proposed with [I-D.ietf-sip-authid-body]. The enhancement is based on the attributes asserted by the Authentication Service.

Figure 3 shows three entities, Alice@example.com, AS@example.com and Bob@example2.com. If Alice wants to communicate with Bob, she sends a SIP INVITE to her preferred AS. Depending on the chosen SIP security mechanism either digest authentication, S/MIME or Transport Layer Security is used to provide the AS with a strong assurance about the identity of Alice. During this step authorization attributes for inclusion into the SAML Header can be selected.

After Alice is authenticated and authorized, a SAML assertion is attached to the SIP message. The Authentication Service can be configured to attach a number of assertions, not limited to the authenticated identity.

To bind the SAML assertion to a specific SIP session, it is necessary for the AS to compute a hash of some fields of the message. A list of the fields to hash is described in [I-D.ietf-sip-identity] and particularly in [I-D.ietf-sip-authid-body]. The hash is digitally signed and inserted into the SAML assertion and placed into the SAML header. The SAML header also contains information about the identity which created the digital signature. Upon reception of the message, Bob learns the signature in the SAML header and verifies the binding to the SIP message in order to prevent cut-and-paste attacks. The provided SAML assertion can then be used to assist during the authorization procedure. If Bob does not understand the extension defined in this document, he silently ignores it. When the 200 OK message arrives at Bob's AS, the same procedure as when Alice sent her INVITE to her AS can be performed, if desired. This exchange is not shown in Figure 3.

Note that this scenario does not imply that the SAML assertions are solely used by SIP UAs. Assertions can also be helpful for SIP

proxies or B2B UAs. Additionally, a push model is shown in this section but it is reasonable to use a pull as well. For simplicity reasons a push model should be prefered since an additional message exchange between the Authentication Service and the UA can be omitted.

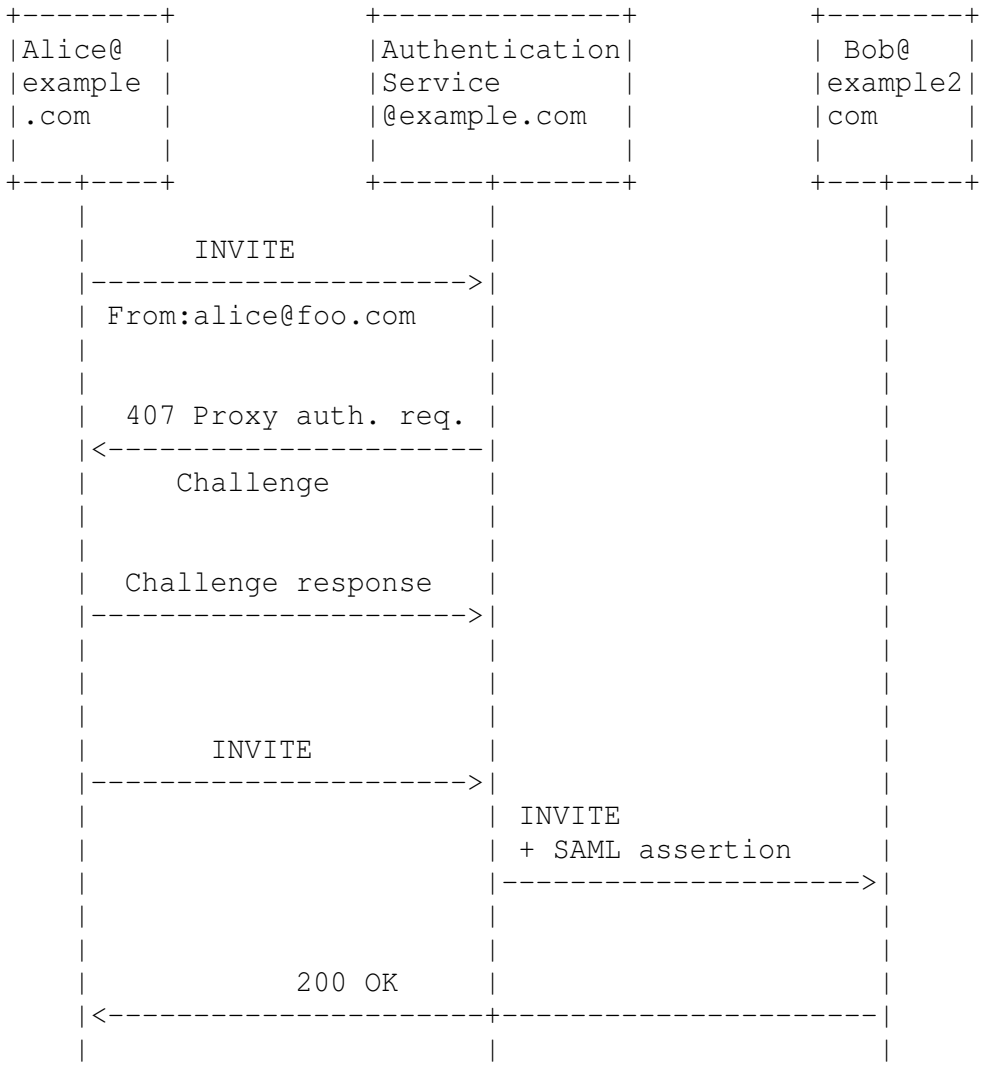


Figure 3: Network Asserted Identities

A variation of the scenario presented in Figure 3 is given in Figure 4 where an end host (Alice@example.com) obtains an Assertion from its SIP proxy server. This assertion is then attached by the end host to the outgoing INVITE message. Unlike in case of an Artifact, Bob@example.com does not need to contact the Proxy Server.

An example of this scenario could be to preempt a lower priority call if enough assurance for doing so is presented in the attached SAML assertion. This would also mean that there is a priority value included in the INVITE (for example in the Resource-Priority Header).

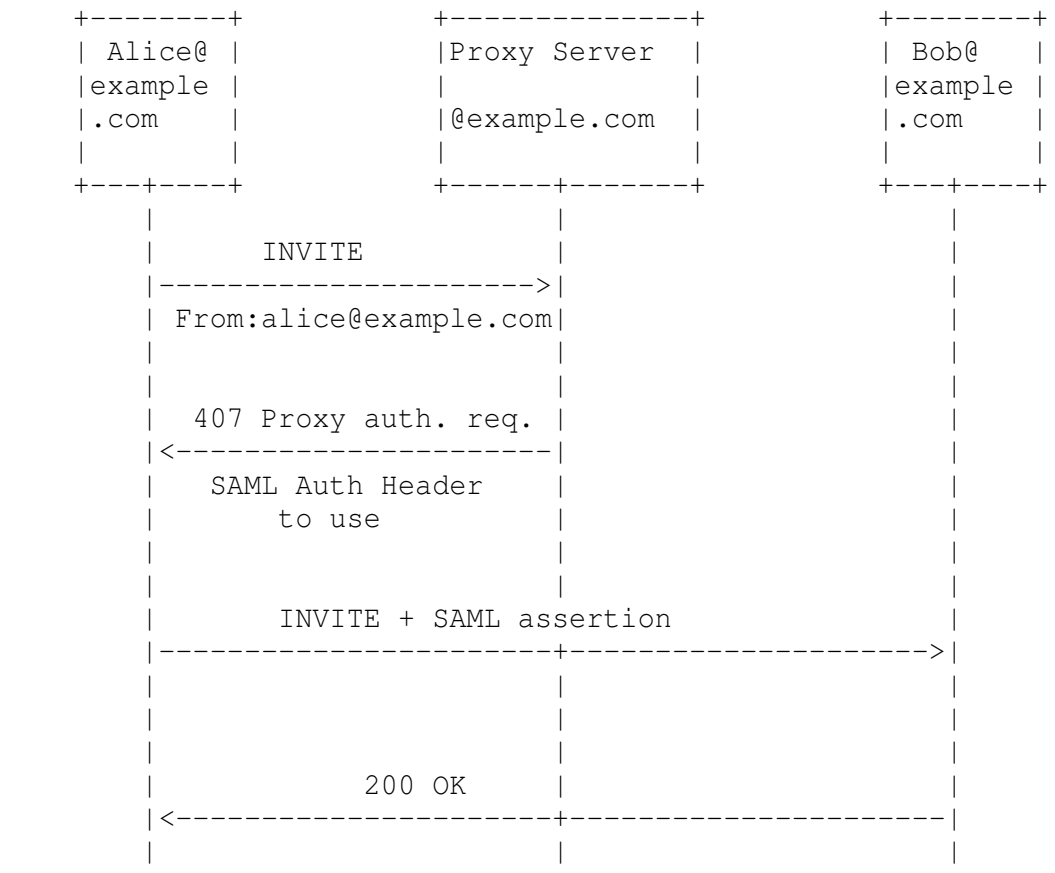


Figure 4: End host attaching SAML Assertion

Note that Bob and Alice can do not need to be in the same administrative domain. It is feasible that Bob is in a domain that is federated with Alice's domain.

The Assertion obtained by Alice@example.com needs to be associated with a particular SIP messaging session. How to achieve this binding is for further consideration.

6.2 SIP Conferencing

This section is meant to raise some discussions about the usage of SAML in the domain of SIP conferencing.

Two possible usage cases are described. The first use case describes a user who routes its SIP message through the Authentication Service (Asserting Party) to allow SAML Assertions to be included. The following goals could be achieved by this procedure:

- The user identity can be replaced to have the user to be anonymous with regard to the Focus
- The user identity could be asserted to the Focus
- The SAML Assertion could provide additional information such as group membership (belongs to the students,staff, faculty group of university X). This could, for non-identity-based authorization system imply certain rights.

The corresponding SIP message flow (in high level detail) could have the following shape:

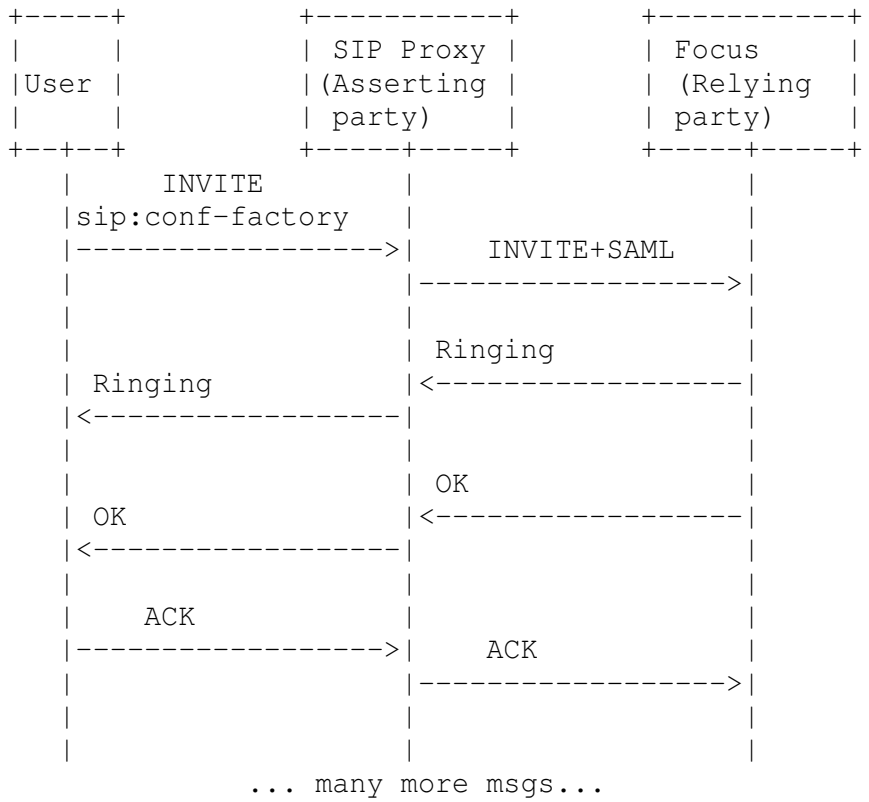


Figure 5: SIP Conferencing and SAML

In the second use case, a SIP Proxy (which acts as an Asserting Party) stores authorization policies for usage with SIP conferencing. These authorization policies could be attached by the Asserting Party

for this particular session. The Focus does not need to store this type of information.

These policies might not be configured by the user itself. One could think of a model similar to the Geopriv model where the rule maker does not need to be the same entity as the conferencing participant.

The authorization policies defined in [I-D.niemi-xcon-cpcp-rules] could be reused. If we think of virtual 3GPP/IEEE meeting scenario then certain members have more rights than others and it must be asserted that a particular person is the member of a company and possibly some other restrictions.

The SAML assertion could, for example, include (among other things):

```
<rule="a1234">
 <conditions/>
 <actions>
   <join-handling>accept</join-handling>
 </actions>
 <transformations>
   <is-key-participant>true</is-key-participant>
 </transformations>
</rule>
```

Figure 6: Carrying authz. policies for SIP Conferering in SAML

The identity section of the conditions part is left empty since the Assertion is attached to a request which originates from a particular entity. The identity to whom the Assertion was granted is already included in other SAML specific attributes.

6.3 PSTN-to-SIP phone call

Alice, using a phone connected to the PSTN, wants to make a call to Bob, which resides in a SIP network. Her call is switched through the PSTN by means of PSTN signaling (outside the scope of this document) to the PSTN/SIP gateway. At the gateway, the call is converted from SS7 signaling to SIP signaling. Since Alice was previously properly authenticated through PSTN signaling mechanisms, the gateway can create an assertion based on signaling information from Alice and digitally sign it with it’s private key. Alice’s call is forwarded from the SIP/PSTN gateway to Bob’s phone. Bob can certify that the identity of Alice is correct by examining the SAML assertion.

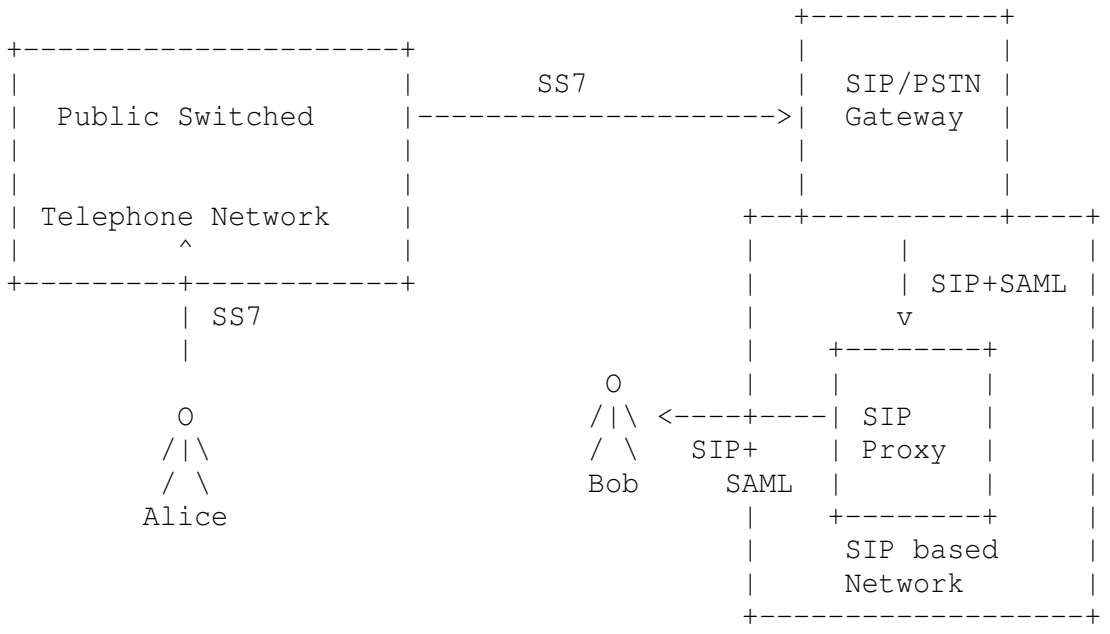


Figure 7: PSTN to SIP call

7. Header Syntax

This document specifies a new SIP header: SAML-Payload

This header MUST NOT appear more than once in a SIP message.

SAML-Payload = "SAML-Payload" HCOLON saml-assertion

The saml-assertion field either contains a base64 encoded SAML Assertion which SHOULD be S/MIME protected or an Artifcat. An example of an Assertion is shown in Section 8 and its protection is highly recommended but not mandated by the SAML specification.

If an SAML assertions is protected then S/MIME MUST be used rather than XML digital signatures.

To bind a SAML assertion to a SIP message a few selected SIP message fields are input to a hash function. The digest-string, defined in Section 10 of [I-D.ietf-sip-identity], is included into the TBD element of the SAML Assertion.

8. Example

This is an example of a SAML assertion and how it is structured in XML.

```
<saml:Assertion
 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
 MajorVersion="1"
 MinorVersion="1"
 AssertionID="P1YaAz/tP6U/fsw/xA+jax5TPxQ="
 Issuer="www.example.com"
 IssueInstant="2004-06-28T17:15:32.753Z">
    <saml:Conditions NotBefore="2004-06-28T17:10:32.753Z"
          NotOnOrAfter="2004-06-28T17:20:32.753Z" />
    <saml:AuthenticationStatement
      AuthenticationMethod="urn:ietf:rfc:3075"
      AuthenticationInstant="2004-06-28T17:15:12.706Z">
           <saml:Subject>
                   <saml:NameIdentifier>
                     NameQualifier=alice@example.com
                     Format="urn:oasis:names:tc:SAML:1.1:nameid-
                      format:emailAddress">uid=alice
                   </saml:NameIdentifier>
                   <saml:SubjectConfirmation>
                          <saml:ConfirmationMethod>
                            urn:oasis:names:tc:SAML:1.0:
                             cm:SIP-artifact-01
                          </saml:ConfirmationMethod>
                   </saml:SubjectConfirmation>
           </saml:Subject>
    </saml:AuthenticationStatement>
</saml:Assertion>
```

The elements in the Assertion have the following meaning:

Tag name	Required	Description
MajorVersion	X	Major version of the assertion
MinorVersion	X	Minor version of the assertion
AssertionID	X	ID of the assertion
Issuer	X	The name of the SAML authority that created the assertion

IssuerInstant	X	Issuing time of the assertion
Conditions		Conditions that MUST be taken into account when assessing the validity of the assertion
AuthenticationMethod	X	A URI reference that specifies what kind of authentication took place
AuthenticationInstant	X	Specifies the time when the authentication took place
Qualifier		The name by which the subject is recognized
Format		A URI reference representing the format of NameIdentifier
SubjectConfirmation		Specifies a subject by supplying data that allows the subject to be authenticated
ConfirmationMethod		A URI reference who identifies which method to be used for authenticating the subject

Figure 9: Tag descriptions

9. Requirement Comparison

A future version of this document will compare the requirements listed in [I-D.ietf-sipping-trait-authz] with the solution provided in this document.

10. Security considerations

This section discusses security considerations when using SAML with SIP.

10.1 Stolen Assertion

Threat: If an eavesdropper can copy the real user's SAML response and included assertions and construct it's own SIP message, then the eavesdropper could be able to impersonate the user at other SIP entities.

Countermeasures: By providing adequate confidentiality, eavesdropping of a SAML assertion can be stopped.

10.2 MitM Attack

Threat: Since the SAML assertion is carried within a SIP message, a malicious site could impersonate the user at some other SIP entities. These SIP entities would believe the adversary to be the subject of the assertion.

Countermeasures: If the adversary is a not-participating in the SIP signaling itself (i.e., it is not a SIP proxy or a SIP UA), this threat can be eliminated by employing inherent SIP security mechanisms, such as TLS. However, if this entity is part of the communication itself then reference integrity needs to be provided. Assertions with tight restrictions (e.g., validity of the assertion) can also limit the possible damage.

10.3 Forged Assertion

Threat: A malicious user could forge or alter a SAML assertion in order to communicate with the SIP entities.

Countermeasures: To avoid this kind of attack, the entities must assure that proper mechanisms for protecting the SAML assertion needs to be in place. It is recommended to protect the Assertion using a digital signature.

10.4 Replay Attack

Threat: In the case of using SIP with the SAML pull model, the threat of replay lies in the fact that the artifact is a one-time-use subject. The same artifact can be used again to gain access to resources.

Countermeasures: Cases where multiple requests are made which

references the same request must be tracked in order to avoid the threat.

11. Contributors

The authors would like to thank Henning Schulzrinne for their input to this document.

12. IANA Considerations

This document contains a number of IANA considerations. A future version of this document will list them in this section.

13. Open issues

During the work on this document a number of open issues have been discovered:

- It needs to be studied how the interworking between reference integrity and the usage of obtained SAML assertion can be properly accomplished.

- In Section 7, it suggests that an S/MIME protected SAML assertion could appear by-value in a SIP header. This issue deserves further investigations. If by-value inclusion is required, then at best, this header might contain a "cid:" URL. This allows you to point to a particular MIME body in the body of a SIP message. However, proxy servers cannot add bodies to SIP requests, and so this would be a UA-only approach.

- Where should the Assertions be attached? At the UA or at the proxy? In the scenarios depicted in Section 6, we have both approaches depending on what kind of scenario it is. In Figure 3, they are added at the UA and in contrast we have Figure 7, where the assertions are added at the PSTN/SIP gateway.

  MIME bodies can only be attached at the UA. Proxies by definition do not attach MIME bodies; if an intermediary were to do so, it would not be playing the proxy server role in the SIP architecture. If an assertion needs to be added by-value, something like the redirection-based mechanism originally in the identity draft would be appropriate. This issue is for further study.

- The usage of the SAML artifact from HTTP to provide by-reference carriage of the assertion needs further study. The HTTP artifact assumes that the recipient will know how to dereference the artifact, which for various reasons in the HTTP binding makes sense, but in SIP might not. The authors think that it would be good to provide a URL associated with the asserting party, one which might also contain the artifact (i.e. http://asserting-party.com/artifact), or parts of the artifact, that are necessary to index to a particular assertion being held by the asserted party. Hence, an HTTPS URL has to be used for this purpose. Otherwise, given the very peer-to-peer nature of SIP it is possible that the artifact will not give you enough information to be able to figure out how to contact the asserting party.

- Some work on option-tags is required. Are there cases when processing of the assertion would be required by the sender? Or

when a proxy server wants to be able to say that a UA must supply this header in order to access a particular resource? If so (and I think so), an option-tag should be defined for this extension that can be used in Require, Supported, 420, etc.

- Specific SAML confirmation method identifiers and identifiers for the bindings or profiles must be defined and registrered with OASIS. A confirmation method identifier is a URI that specifies which method should be used by the target domain to assure that the identity of the subject is true.

  This mechanism seems to be provide the same reference integrity properties as the hash over the various headers/bodies proposed in the identity draft.

- A few new URIs need to be registered. The proposed URIs for identification are:

  * SIP Binding: urn:oasis:names:tc:SAML:1.0:bindings:SIP-binding

  * Artifact profile: urn:oasis:names:tc:SAML:1.0:profiles:SIP-artifact-01

  * Assertion profile: urn:oasis:names:tc:SAML:1.0:profiles:SIP-assertion-01

- The proposed URIs for Confirmation Method Identifiers are:

  * Artifact profile: urn:oasis:names:tc:SAML:1.0:cm:SIP-artifact-01

  * Assertion profile: urn:oasis:names:tc:SAML:1.0:cm:SIP-bearer

- These are based on the URIs already used in the existing SOAP-SAML binding, specified in Section 3 of [I-D.saml-bindings-1.1].

- An alignment with the work done by Liberty Alliance on Federated Identities as described in [I-D.liberty-idff-arch-overview] would be useful.

- The security consideration needs more details.

14. References

14.1 Normative References

[I-D.ietf-sipping-trait-authz]
Peterson, J., "Trait-based Authorization Requirements for the Session Initiation Protocol (SIP)", draft-ietf-sipping-trait-authz-00 (work in progress), February 2004, <reference.I-D.ietf-sipping-trait-authz.xml>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

14.2 Informative References

[I-D.draft-ietf-simple-xcap-02]
Rosenberg, J., "Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-02 (work in progress), February 2004, <reference.I-D.draft-ietf-simple-xcap-02.xml>.

[I-D.ietf-sip-authid-body]
Peterson, J., "SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-03 (work in progress), May 2004, <reference.I-D.ietf-sip-authid-body.xml>.

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

[I-D.liberty-idff-arch-overview]
Wason, T., "Liberty ID-FF Architecture Overview", 2004, <reference.I-D.liberty-idff-arch-overview.xml>.

[I-D.niemi-xcon-cpcp-rules]
Niemi, A., "Conference Policy Authorization Rules", draft-niemi-xcon-cpcp-rules-00 (work in progress), May 2004, <reference.I-D.niemi-xcon-cpcp-rules.xml>.

[I-D.saml-bindings-1.1]
Maler, E., Philpott, R. and P. Mishra, "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1", September 2003, <reference.I-D.saml-bindings-1.1.xml>.

[I-D.saml-core-1.1]
Maler, E., Philpott, R. and P. Mishra, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1", September 2003, <reference.I-D.saml-core-1.1.xml>.

[I-D.saml-sec-consider-1.1]
Maler, E. and R. Philpott, "Security and Privacy Considerations for the OASIS Security Markup Language (SAML) V1.1", September 2003, <reference.I-D.saml-sec-consider-1.1.xml>.

[I-D.saml-tech-overview-1.1-03]
Maler, E. and J. Hughes, "Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1", March 2004, <reference.I-D.saml-tech-overview-1.1-03.xml>.

[RFC2543] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999, <reference.RFC.2543.xml>.

Authors' Addresses

Hannes Tschofenig
Siemens
Otto-Hahn-Ring 6
Munich, Bayern 81739
Germany

EMail: Hannes.Tschofenig@siemens.com

Jon Peterson
NeuStar, Inc.
1800 Sutter St Suite 570
Concord, CA 94520
US

EMail: jon.peterson@neustar.biz

James Polk
Cisco
2200 East President George Bush Turnpike
Richardson, Texas 75082
US

EMail: jmpolk@cisco.com

Douglas C. Sicker
University of Colorado at Boulder
ECOT 531
Boulder, CO 80309
US

EMail: douglas.sicker@colorado.edu

Marcus Tegnander
Siemens
Otto-Hahn-Ring 6
Munich, Bayern 81739
Germany

Appendix A. SAML Artifact format

```
TypeCode           := 0x0001
RemainingArtifact  := SourceID AssertionHandle
SourceID           := 20-byte_sequence
AssertionHandle    := 20-byte_sequence
```

The sourceID is a 20-byte sequence which is used by the destination domain to determine the identity and location of the source domain. It is assumed that the destination domain will maintain a table of sourceID values as well as URLs for the domains it is having SAML sessions with and this information is communicated out-of-band. When the destination receives the SAML artifact, it determines if the source domain belongs to a known source domain and obtains the location before sending a SAML request.

If two source domains communicating with the same target domain MUST be distinguished by unique SourceIDs. The AssertionHandle is constructed on the basis that they SHOULD not have any predictable relationship to the contents of the assertion and that it MUST be not worth the effort to reconstruct or guess the value of the AssertionHandle.

In [I-D.saml-bindings-1.1] section 4.1.1.8, some RECOMMENDED practices for creation of SAML artifacts are listed.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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