SIPP
Internet-Draft
Expires: October 3, 2005

J. Rosenberg
Cisco Systems
April 4, 2005

Request Authorization through Dialog Identification in the Session Initiation Protocol (SIP)
draft-ietf-sip-target-dialog-00

Status of this Memo

Copyright Notice

Abstract

This specification defines the Target-Dialog header field for the Session Initiation Protocol (SIP), and the corresponding option tag, tdialog. This header field is used in requests that create SIP dialogs. It indicates to the recipient that the sender is aware of an existing dialog with the recipient, either because the sender is on the other side of that dialog, or because it has access to the dialog identifiers. The recipient can then authorize the request

based on this awareness.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1] defines the concept of a dialog as a persistent relationship between a pair of user agents. Dialogs provide context, including sequence numbers, proxy routes, and dialog identifiers. Dialogs are established through the transmission of SIP requests with particular methods. Specifically, the INVITE, REFER [7], SUBSCRIBE and NOTIFY [2] requests all create dialogs.

When a user agent receives a request that creates a dialog, it needs to decide whether to authorize that request. For some requests, authorization is a function of the identity of the sender, the request method, and so on. However, many situations have been identified in which a user agents' authorization decision depends on whether the sender of the request is currently in a dialog with it, or aware of a dialog with it.

One such example is call transfer, accomplished through REFER. If user agents A and B are in an INVITE dialog, and user agent A wishes to transfer user agent B to user agent C, user agent A needs to send a REFER request to user agent B, asking user agent B to send an INVITE request to user agent C. User agent B needs to authorize this REFER. The proper authorization decision is that user agent B should accept the request if it came from a user with whom B currently has an INVITE dialog relationship. Current implementations deal with this by sending the REFER on the same dialog as the one in place between user agents A and B. However, this approach has numerous problems [8]. These problems include difficulty in determining the lifecycle of the dialog and its usages and difficulties in determining which messages are associated with each application usage. Instead, a better approach is for user agent A to send the REFER request to user agent C outside of the dialog using its Globally Routable User Agent URI (GRUU) [9]. In that case, a means is needed for user agent B to authorize the REFER.

Another example is the application interaction framework [10]. In that framework, proxy servers on the path of a SIP INVITE request can place user interface components on the user agent that generated or received the request. To do this, the proxy server needs to send a REFER request to the user agent, targeted to their GRUU, asking the user agent to fetch an HTTP resource containing the user interface. In such a case, a means is needed for the user agent to authorize the REFER.

Another example is if two user agents share an INVITE dialog, and an element on the path of the INVITE request wishes to track the state of the INVITE. In such a case, it sends a SUBSCRIBE request to the

GRUU of the user agent, asking for a subscription to the dialog event package. If the SUBSCRIBE request came from an element on the INVITE request path, it should be authorized.

2. Overview of Operation

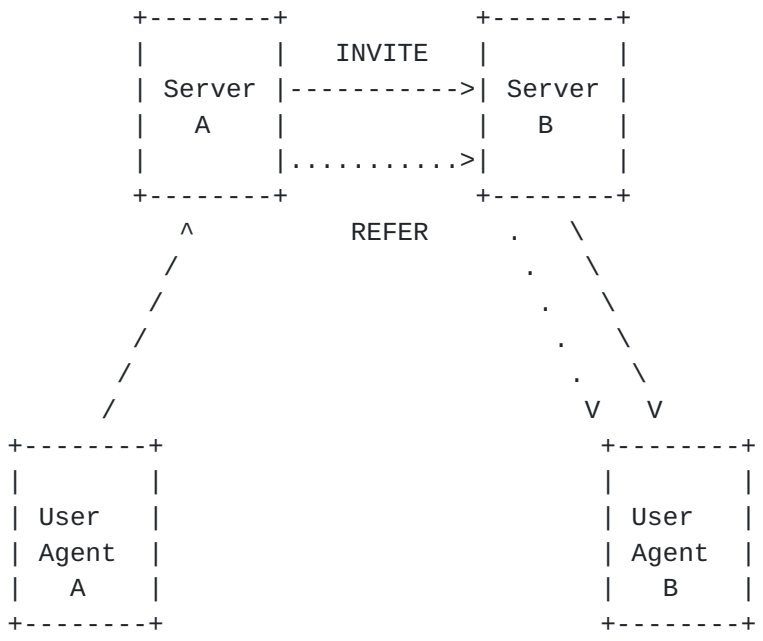

Figure 1

Figure 1 shows the basic model of operation. User agent A sends an INVITE to user agent B, traversing two servers, server A and server B. Both servers act as proxies for this transaction. User B sends a 200 OK response to the INVITE. This 200 OK includes a Supported header field indicating support for both the GRUU specification (through the presence of the gruu option tag) and this specification (through the presence of the tdialog option tag). The 200 OK response establishes a dialog between the two user agents. Next, server A wishes to REFER user agent B to fetch an HTTP resource. So, it acts as a user agent and sends a REFER request to user agent B. This REFER is addressed to the GRUU of user agent B, which server A learned from inspecting the Contact header field in the 200 OK of the INVITE request. This GRUU is a URI that can be used by any element on the Internet, such as server A, to reach the specific user agent

instance that generated that 200 OK to the INVITE.

The REFER request generated by server A will contain a Target-Dialog header field. This header field contains the dialog identifiers for the INVITE dialog between user agents A and B, composed of the Call-ID, local tag, and remote tag. Server A knew to include the Target-Dialog header field in the REFER request because it knows that user agent B supports it.

When the REFER request arrives at user agent B, it needs to make an authorization decision. Because the INVITE dialog was established using a sips URI, and because the dialog identifiers are cryptographically random [1], no entity except for user agent A or the proxies on the path of the initial INVITE request can know the dialog identifiers. Thus, because the REFER request contains those dialog identifiers, user agent B can be certain that the REFER request came from either user agent A, the two proxies, or an entity to whom the user agent or proxies gave the dialog identifiers. As such, it authorizes the REFER request, and fetches the HTTP resource identified by the URI of the Refer-To header field in the REFER request.

3. UAC Behavior

A UAC SHOULD include a Target-Dialog header field in a request if the following conditions are all true:

1. The request is to be sent outside of any existing dialog.

2. The user agent client believes that the request will not be authorized by the user agent server unless the user agent client can prove that it is aware of the dialog identifiers for some other dialog. Call this dialog the target dialog. Examples of this condition include (1) REFER requests sent outside of a dialog by a participant in the dialog, (2) subscriptions to the dialog event package [11] when that SUBSCRIBE request is sent by a participant in the dialog, and (3) subscriptions to the conference event package [12] when that SUBSCRIBE request is sent by a participant in the conference with a dialog to the focus.

3. The user agent client knows that the user agent server supports the Target-Dialog header field. It can know this if it has seen a request or response from the user agent server within the target dialog that contained a Supported header field which included the tdialog option tag.

If the third condition is not met, the UAC SHOULD NOT use this specification. Instead, if it is currently within a dialog with the

UAS, it SHOULD attempt to send the request within the existing target dialog.

The value of the call-id production in the Target-Dialog header field MUST be equal to the Call-ID of the target dialog. The "remote-tag" header field parameter MUST be present, and MUST contain the tag that would be viewed as the remote tag from the perspective of the recipient of the new request. The "local-tag" header field parameter MUST be present, and MUST contain the tag that would be viewed as the local tag from the perspective of the recipient of the new request.

The request sent by the UAC SHOULD include a Require header field that includes the tdialog option tag. This request should, in principle, never fail with a 420 (Bad Extension) response, because the UAC would not have sent the request unless it believed the UAS supported the extension. If a Require header field was not included, and the UAS didn't support the extension, it would normally reject the request becaust it was unauthorized, probably with a 403. However, without the Require header field, the UAC would not be able to differentiate a 403 that arrived because the UAS didn't actually understand the Target-Dialog header field (in which case the client should send the request within the target dialog if it can), from a 403 that arrived because the UAS understood the Target-Dialog header field, but elected not to authorize the request despite the fact that the UAC proved its awareness of the target dialog (in which case the client should not resend the request within the target dialog, even if it could).

4. User Agent Server Behavior

If a user agent server receives a dialog-creating request, and wishes to authorize the request, and that authorization depends on whether or not the sender has knowledge of an existing dialog with the UAS, the UAS SHOULD check the request for the existence of the Target-Dialog header field. If this header field is not present, the UAS MAY still authorize the request based on other means.

If the header field is present, and the value of the call-id production, the "remote-tag" and "local-tag" values match the Call-ID, remote tag and local tag of an existing dialog, and the dialog that they match was established using a sips URI, the UAS SHOULD authorize the request if it would authorize any entity on the path of the request that created that dialog, or any entity trusted by an entity on the path of the request that created that dialog.

If the dialog identifiers match, but they match a dialog not created with a sips URI, the UAS MAY authorize the request if it would authorize any entity on the path of the request that created that

dialog, or any entity trusted by an entity on the path of the request that created that dialog. However, in this case, any eavesdropper on the original dialog path would have access to the dialog identifiers, and thus the authorization strength is reduced to MAY.

If the dialog identifiers don't match, or if they don't contain both a "remote-tag" and "local-tag" parameter, the header field MUST be ignored, and authorization MAY be determined by other means.

5. Proxy Behavior

Proxy behavior is unaffected by this specification.

6. Extensibility Considerations

This specification depends on a user agent client knowing, ahead of sending a request to a user agent server, whether or not that user agent server supports the Target-Dialog header field. As discussed in Section 3, the UAC can know this because it saw a request or response sent by that UAS within the target dialog that contained the Supported header field whose value included the tdialog option tag.

Because of this requirement, it is especially important that user agents compliant to this specification include a Supported header field in all dialog forming requests and responses. Inclusion of the Supported header fields in requests is at SHOULD strength within RFC 3261. This specification does not alter that requirement. However, implementors should realize that, unless the tdialog option tag is placed in the Supported header field of requests and responses, this extension is not likely to be used, and instead, the request is likely to be resent within the existing target dialog (assuming the sender is the UA on the other side of the target dialog). As such, the conditions in which the SHOULD would not be followed would be those rare cases in which the UA does not want to enable usage of this extension.

7. Header Field Definition

The grammar for the Target-Dialog header field is defined as follows:

```
Target-Dialog      =     "Target-Dialog" HCOLON call-id *(SEMI
                         td-param)
td-param           =     remote-param / local-param / generic-param
remote-param       =     "remote-tag" EQUAL token
local-param        =     "local-tag" EQUAL token
```

Figure 3 and Figure 4 are an extension of Tables 2 and 3 in RFC 3261

[1] for the Target-Dialog header field. The column "INF" is for the INFO method [3], "PRA" is for the PRACK method [4], "UPD" is for the UPDATE method [5], "SUB" is for the SUBSCRIBE method [2], "NOT" is for the NOTIFY method [2], "MSG" is for the MESSAGE method [6], and "REF" is for the REFER method [7]

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Target-Dialog	R	ar	-	-	-	o	-	-

Figure 3: Allowed Methods for Target-Dialog

Header field	where	proxy	PRA	UPD	SUB	NOT	INF	MSG	REF
Target-Dialog	R	ar	-	-	o	-	-	-	o

Figure 4: Allowed Methods for Target-Dialog

8. Security Considerations

The Target-Dialog header field is used to authorize requests based on the fact that the sender of the request has access to information that only certain entities have access to. In order for such an authorization decision to be secure, two conditions have to be met. Firstly, no eavesdroppers can have access to this information. That requires the original SIP dialog to be established using a sips URI, which provides TLS on each hop. With a sips URI, only the user agents and proxies on the request path will be able to know the dialog identifiers. The second condition is that the dialog identifiers be sufficiently random that they cannot be guessed. RFC 3261 requires global uniquess for the Call-ID and 32 bits of randomness for each tag (there are two tags for a dialog). Given the short duration over which a typical dialog exists (perhaps as long as a day), this amount of randomness appears adequate to prevent guessing attacks.

9. Example Call Flow

In this example, user agent A and user agent B establish an INVITE initiated dialog. User agent A would then like to establish a subscription to the dialog event package. To do this, it sends a SUBSCRIBE request outside of the context of the dialog, using the Target-Dialog header field. This call flow is shown in Figure 5.

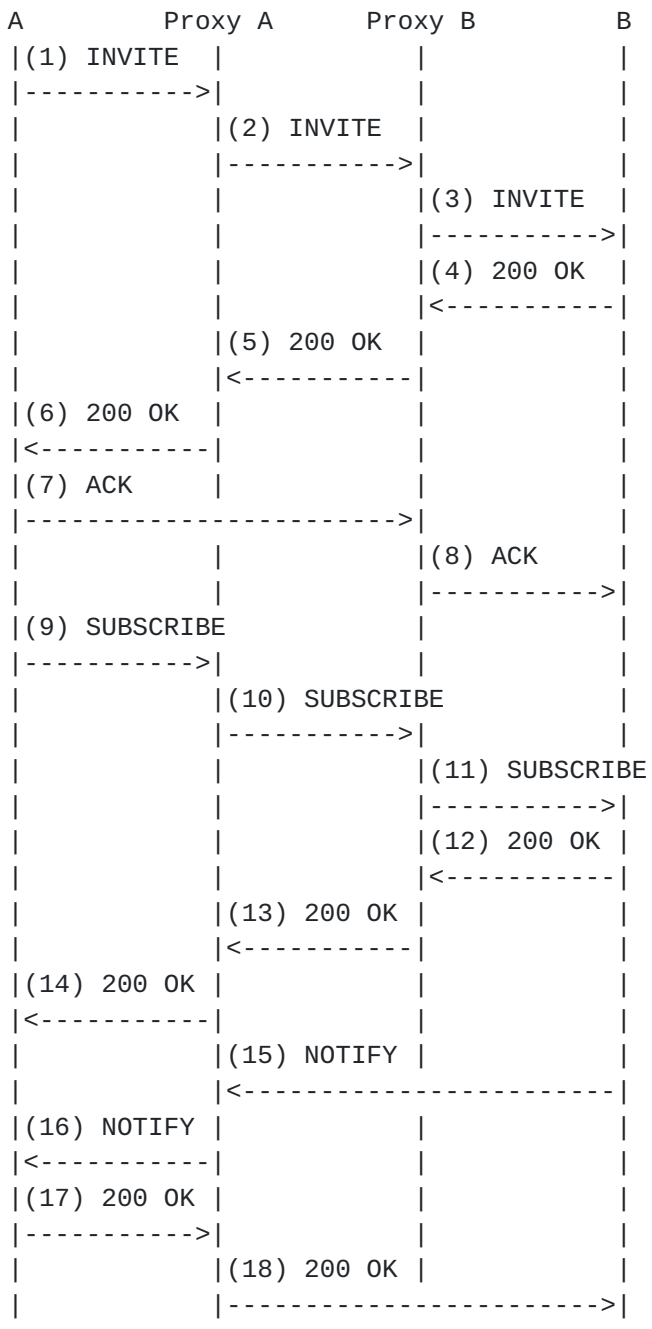

Figure 5

First, the caller sends an INVITE, as shown in message 1.

```
INVITE sips:B@example.com SIP/2.0
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.org>
Call-ID: fa77as7dad8-sd98ajzz@host.example.com
CSeq: 1 INVITE
Max-Forwards: 70
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, NOTIFY
Accept: application/dialog-info+xml
Contact: <sips:bad998asd8asd0000a@example.com>
Content-Length: ...
Content-Type: application/sdp

--SDP not shown--
```

This is forwarded to the callee (messages 2-3), which generates a 200 OK response that is forwarded back to the caller (message 4-5). The 200 OK received by the caller (message 6) will look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8;received=192.0.2.1
From: Caller <sip:A@example.com>;tag=kkaz-
To: Callee <sip:B@example.org>;tag=6544
Call-ID: fa77as7dad8-sd98ajzz@host.example.com
CSeq: 1 INVITE
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, SUBSCRIBE
Allow-Events: dialog
Contact: <sips:hgasd9f88ggd7gasl@example.org>
Content-Length: ...
Content-Type: application/sdp

--SDP not shown--
```

The caller generates an ACK (message 7). Because user B made use of a GRUU, even though neither proxy record-routed, the ACK is delivered to the proxy of user B, and then forwarded to them (message 8). User A then decides to subscribe to the dialog event package for dialog events at user B. The SUBSCRIBE it sends looks like:

```
SUBSCRIBE sips:hgasd9f88ggd7gasl@example.org SIP/2.0
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz10
From: Caller <sip:A@example.com>;tag=mreysh
To: Callee <sips:hgasd9f88ggd7gasl@example.org>
Event: dialog;call-id=fa77as7dad8-sd98ajzz@host.example.com
  ;from-tag=kkaz-;to-tag=6544
Accept: application/dialog-info+xml
Target-Dialog: fa77as7dad8-sd98ajzz@host.example.com
  ;remote-tag=kkaz-
  ;local-tag=6544
Call-ID: 86d65asfklzll8f7asdr@host.example.com
CSeq: 1 SUBSCRIBE
Max-Forwards: 70
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, NOTIFY
Contact: <sips:bad998asd8asd0000a@example.com>;schemes="sip,sips"
Content-Length: 0
```

Note that this SUBSCRIBE request contains both a Target-Dialog header field and Event header field parameters that identify the dialog being subscribed to. Those these have the same content (dialog identifers for the existing INVITE dialog), they have different semantics. The Target-Dialog header field is used for authorization, and the Event header field parameters indicate which dialogs are being subscribed to. These need not be the same.

The SUBSCRIBE is forwarded to proxy B (message 10), and from there to user B (message 11). Because of the presence of the Target-Dialog header field, the request is authorized and processed. It responds with a 200 OK (message 12), which is forwarded to proxy A (message 13) and then to user A (message 14). This response looks like:

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz10;received=192.0.2.1
From: Caller <sip:A@example.com>;tag=mreysh
To: Callee <sips:hgasd9f88ggd7gasl@example.org>;tag=trdppmdrysh
Call-ID: 86d65asfklzll8f7asdr@host.example.com
CSeq: 1 SUBSCRIBE
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, SUBSCRIBE
Allow-Events: dialog
Contact: <sips:hgasd9f88ggd7gasl@example.org>
Content-Length: 0
```

This is followed by a NOTIFY (message 15).

10. IANA Considerations

This specification registers a new SIP header field and a new option tag according to the processes of RFC 3261 [1].

10.1 Header Field

RFC Number: RFC XXXX [Note to IANA: Fill in with the RFC number of this specification.]

Header Field Name: Target-Dialog

Compact Form: none

10.2 SIP Option Tag

This specification registers a new SIP option tag per the guidelines in Section 27.1 of RFC 3261.

Name: tdialog

Description: This option tag is used to identify the target dialog header field extension. When used in a Require header field, it implies that the recipient needs to support the Target-Dialog header field. When used in a Supported header field, it implies that the sender of the message supports it.

11. Acknowledgments

This specification is based on a header field first proposed by Robert Sparks in the dialog usage draft. John Elwell provided helpful comments.

12. References

12.1 Normative References

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

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

[3] Donovan, S., "The SIP INFO Method", RFC 2976, October 2000.

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

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

[6] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C. and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

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

12.2 Informative References

[8] Sparks, R., "Multiple Dialog Usages in the Session Initiation Protocol", draft-sparks-sipping-dialogusage-00 (work in progress), July 2004.

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

[10] Rosenberg, J., "A Framework for Application Interaction in the Session Initiation Protocol (SIP)", draft-ietf-sipping-app-interaction-framework-04 (work in progress), February 2005.

[11] Rosenberg, J., "An INVITE Inititiated Dialog Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-dialog-package-05 (work in progress), November 2004.

[12] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Conference State", draft-ietf-sipping-conference-package-08 (work in progress), December 2004.

Author's Address

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

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

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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