

Network Working Group  
Request for Comments: 4538  
Category: Standards Track

J. Rosenberg  
Cisco Systems  
June 2006

Request Authorization through Dialog Identification  
in the Session Initiation Protocol (SIP)

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

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 .....                     | 3  |
| 1.1. Terminology .....                    | 4  |
| 2. Overview of Operation .....            | 4  |
| 3. User Agent Client (UAC) Behavior ..... | 5  |
| 4. User Agent Server Behavior .....       | 7  |
| 5. Proxy Behavior .....                   | 8  |
| 6. Extensibility Considerations .....     | 8  |
| 7. Header Field Definition .....          | 9  |
| 8. Security Considerations .....          | 9  |
| 9. Relationship with In-Reply-To .....    | 10 |
| 10. Example Call Flow .....               | 10 |
| 11. IANA Considerations .....             | 13 |
| 11.1. Header Field .....                  | 13 |
| 11.2. Header Field Parameters .....       | 13 |
| 11.2.1. local-tag .....                   | 13 |
| 11.2.2. remote-tag .....                  | 13 |
| 11.3. SIP Option Tag .....                | 14 |
| 12. Acknowledgements .....                | 14 |
| 13. References .....                      | 14 |
| 13.1. Normative References .....          | 14 |
| 13.2. Informative References .....        | 15 |

## 1. Introduction

The Session Initiation Protocol (SIP) [2] 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 [8], and SUBSCRIBE [3] 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 agent's authorization decision depends on whether the sender of the request is currently in a dialog with that user agent, or whether the sender of the request is aware of a dialog the user agent has with another entity.

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 [12]. These problems include difficulties in determining the lifecycle of the dialog and its usages and 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 B outside of the dialog. In that case, a means is needed for user agent B to authorize the REFER.

Another example is the application interaction framework [14]. 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 its Globally Routable User Agent URI (GRUU) [13], asking the user agent to fetch an HTTP resource containing the user interface component. In such a case, a means is needed for the user agent to authorize the REFER. The application interaction framework recommends that the request be authorized if it was sent from an entity on the path of the original dialog. This can be done by including the dialog identifiers in the

REFER, which prove that the user agent that sent the REFER is aware of those dialog identifiers (this needs to be secured against eavesdroppers through the sips mechanism, of course).

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.

### 1.1. 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 [1].

## 2. Overview of Operation

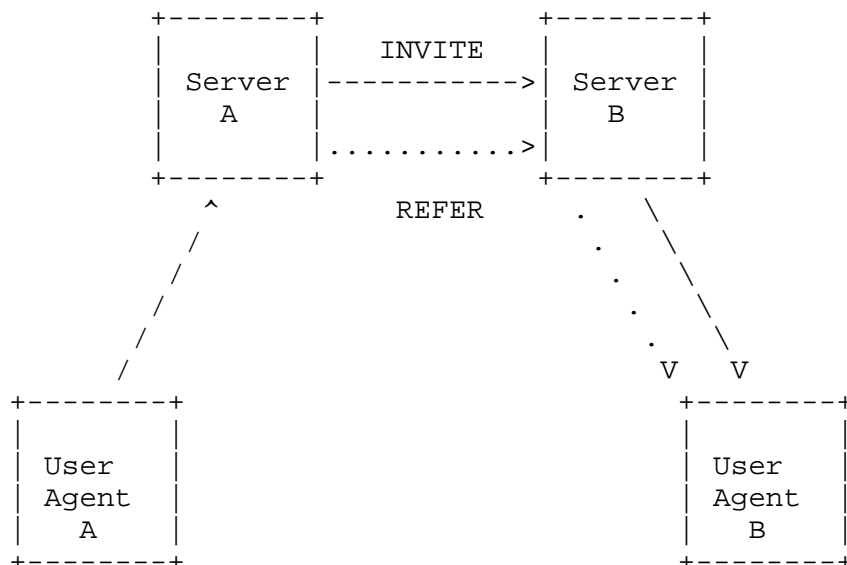


Figure 1

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 this specification (through the presence of the tdialog option tag). The 200 OK response establishes a dialog between the two user agents.

Next, an entity that was present along the request path (server A, for example) wishes to send a dialog-forming request (such as REFER) to user agent A or B (user B for example). So, the entity acts as a user agent and sends the request to user agent B. This request is addressed to the URI of user agent B, which server A learned from inspecting the Contact header field in the 200 OK of the INVITE request. If this URI has the GRUU [11] property (it 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), then the mechanism will work across NAT boundaries.

The 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 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 [2], 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 request contains those dialog identifiers, user agent B can be certain that the request came from 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 request and performs the requested actions.

### 3. User Agent Client (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 may 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.
3. The request does not otherwise contain information that indicates that the UAC is aware of those dialog identifiers.

4. 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 that included the tdialog option tag.

If the fourth condition is not met, the UAC SHOULD NOT use this specification. Instead, if it is currently within a dialog with the User Agent Server (UAS), it SHOULD attempt to send the request within the existing target dialog.

The following are examples of use cases in which these conditions are met:

- o A REFER request is sent according to the principles of [14]. These REFER are sent outside of a dialog and do not contain any other information that indicates awareness of the target dialog. [14] also mandates that the REFER be sent only if the UA indicates support for the target dialog specification.
- o User A is in separate calls with users B and C. User A decides to start a three way call, and so morphs into a focus [17]. User B would like to learn the other participants in the conference. So, it sends a SUBSCRIBE request to user A (who is now acting as the focus) for the conference event package [16]. It is sent outside of the existing dialog between user B and the focus, and it would be authorized by A if user B could prove that it knows the dialog identifiers for its existing dialog with the focus. Thus, the Target-Dialog header field would be included in the SUBSCRIBE.

The following are examples of use cases in which these conditions are not met:

- o A server acting as a proxy is a participant in an INVITE dialog that establishes a session. The server would like to use the Keypad Markup Language (KPML) event package [18] to find out about keypresses from the originating user agent. To do this, it sends a SUBSCRIBE request. However, the Event header field of this SUBSCRIBE contains event parameters that indicate the target dialog of the subscription. As such, the request can be authorized without additional information.
- o A server acting as a proxy is a participant in an INVITE dialog that establishes a session. The server would like to use the dialog event package [15] to find out about dialogs at the originating user agent. To do this, it sends a SUBSCRIBE request. However, the Event header field of this SUBSCRIBE contains event parameters that indicate the target dialog of the subscription.

As such, the request can be authorized without additional information.

Specifications that intend to make use of the Target-Dialog header field SHOULD discuss specific conditions in which it is to be included.

Assuming it is to be included, the value of the callid 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 because it was unauthorized, probably with a 403. However, without the Require header field, the UAC would not be able to differentiate between the following:

- o 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)
- o 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 if that authorization depends on whether or not the sender has knowledge of an existing dialog with the UAS, and information outside of the Target-Dialog header field does not provide proof of this knowledge, 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 by other means.

If the header field is present, and the value of the callid 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 is optional.

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 per RFC 3261. This specification does not alter that requirement. However, implementers 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 re-sent 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 callid *(SEMI
                      td-param) ;callid from RFC 3261
td-param           = remote-param / local-param /
                      generic-param
remote-param       = "remote-tag" EQUAL token
local-param        = "local-tag" EQUAL token
                      ;token and generic-param from RFC 3261

```

Figures 3 and 4 are an extension of Tables 2 and 3 in RFC 3261 [2] for the Target-Dialog header field. The column "INF" is for the INFO method [4], "PRA" is for the PRACK method [5], "UPD" is for the UPDATE method [6], "SUB" is for the SUBSCRIBE method [3], "NOT" is for the NOTIFY method [3], "MSG" is for the MESSAGE method [7], "REF" is for the REFER method [8], and "PUB" is for the PUBLISH method [9].

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

Figure 3: Allowed Methods for Target-Dialog

| Header field  | where | proxy | PRA | UPD | SUB | NOT | INF | MSG | REF |
|---------------|-------|-------|-----|-----|-----|-----|-----|-----|-----|
| Target-Dialog | R     | -     | -   | -   | 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 cryptographically random that they cannot be guessed. RFC 3261 requires global uniqueness for the Call-ID and 32 bits of cryptographic randomness for each tag (there are two tags for a dialog). Given the short duration of a typical dialog (perhaps as long as a day), this amount of randomness appears adequate for

preventing guessing attacks. However, it's important to note that this specification requires true cryptographic randomness as set forth in RFC 4086 [11]. Weaker pseudorandom identifiers reduce the probability of collision, but because they are guessable, they are not sufficient to prevent an attacker from observing a sequence of identifiers, guessing the next one, and then using this specification to launch an attack.

## 9. Relationship with In-Reply-To

RFC 3261 defines the In-Reply-To header field. It provides a list of Call-IDs for calls that the current request references or returns. It was meant to serve a similar purpose as the Reply-To in email: to facilitate the construction of "threads" of conversations in a user interface. Target-Dialog is similar, in that it also references a previous session. Due to their similarities, it is important to understand the differences, as these two header fields are not substitutes for each other.

Firstly, In-Reply-To is meant for consumption by a human or a user interface widget, for providing the user with a context that allows them to decide what a call is about and whether they should take it. Target-Dialog, on the other hand, is meant for consumption by the user agent itself, to facilitate authorization of session requests in specific cases where authorization is not a function of the user, but rather the underlying protocols. A UA will authorize a call containing Target-Dialog based on a correct value of the Target-Dialog header field.

Secondly, Target-Dialog references a specific dialog that must be currently in progress. In-Reply-To references a previous call attempt, most likely one that did not result in a dialog. This is why In-Reply-To uses a Call-ID, and Target-Dialog uses a set of dialog identifiers.

Finally, In-Reply-To implies cause and effect. When In-Reply-To is present, it means that the request is being sent because of the previous request that was delivered. Target-Dialog does not imply cause and effect, merely awareness for the purposes of authorization.

## 10. Example Call Flow

In this example, user agent A and user agent B establish an INVITE-initiated dialog through Server-A and Server-B, each of which acts as a proxy for the INVITE. Server B would then like to use the application interaction framework [14] to request that user agent A fetch an HTML user interface component. To do that, it sends a REFER request to A's URI. The flow for this is shown in Figure 5. The

conventions of [19] are used to describe representation of long message lines.

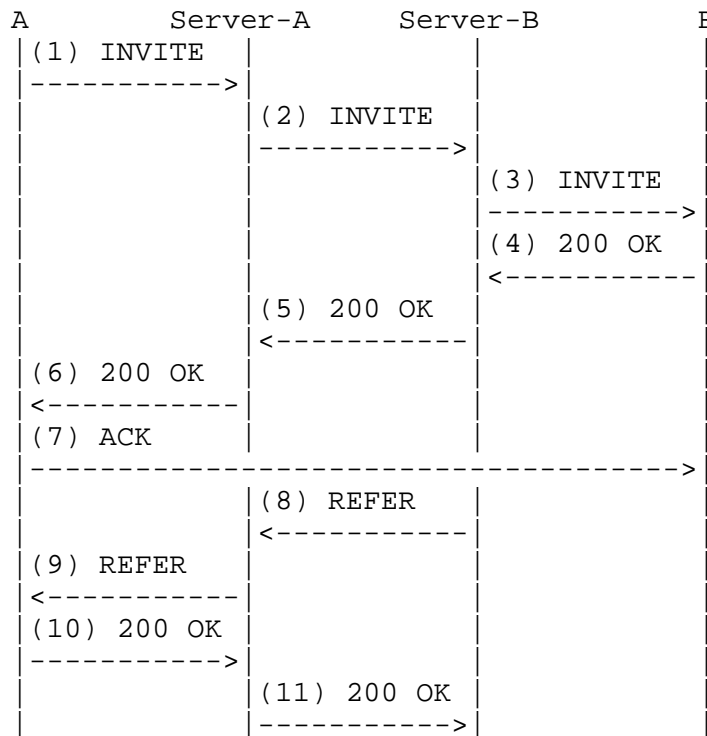


Figure 5

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

```

INVITE sip: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: tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Accept: application/sdp, text/html
<allOneLine>
Contact: <sips:A@example.com;gruu;opaque=urn:uuid:f81d4f
ae-7dec-11d0-a765-00a0c91e6bf6;grid=99a>;schemes="http,sip,sips"
</allOneLine>
Content-Length: ...
Content-Type: application/sdp
  
```

--SDP not shown--

The INVITE indicates that the caller supports GRUU (note its presence in the Contact header field of the INVITE) and the Target-Dialog header field. This INVITE is forwarded to the callee (messages 2-3), which generates a 200 OK response that is forwarded back to the caller (message 4-5). Message 5 might look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS host.example.com;branch=z9hG4bK9zz8
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
Contact: <sips:B@pc.example.org>
Content-Length: ...
Content-Type: application/sdp
```

--SDP not shown--

In this case, the called party does not support GRUU or the Target-Dialog header field. The caller generates an ACK (message 7). Server B then decides to send a REFER to user A:

```
<allOneLine>
REFER sips:A@example.com;gruu;opaque=urn:uuid:f81d4f
ae-7dec-11d0-a765-00a0c91e6bf6;grid=99a SIP/2.0
</allOneLine>
Via: SIP/2.0/TLS serverB.example.org;branch=z9hG4bK9zz10
From: Server B <sip:serverB.example.org>;tag=mreysh
<allOneLine>
To: Caller <sips:A@example.com;gruu;opaque=urn:uuid:f81d4f
ae-7dec-11d0-a765-00a0c91e6bf6;grid=99a>
</allOneLine>
Target-Dialog: fa77as7dad8-sd98ajzz@host.example.com
;local-tag=kkaz-
;remote-tag=6544
Refer-To: http://serverB.example.org/ui-component.html
Call-ID: 86d65asfklz1l8f7asdr@host.example.com
CSeq: 1 REFER
Max-Forwards: 70
Require: tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, NOTIFY
Contact: <sips:serverB.example.org>
Content-Length: 0
```

This REFER will be delivered to server A because it was sent to the GRUU. From there, it is forwarded to user agent A (message 9) and authorized because of the presence of the Target-Dialog header field.

## 11. IANA Considerations

This specification registers a new SIP header field, a new option tag according to the processes of RFC 3261 [2], and two new header field parameters according to the processes of RFC 3968 [10].

### 11.1. Header Field

RFC Number: RFC 4538

Header Field Name: Target-Dialog

Compact Form: none

### 11.2. Header Field Parameters

This section registers two header field parameters according to the processes of RFC 3968 [10].

#### 11.2.1. local-tag

Header Field: Target-Dialog

Header Field Parameter: local-tag

Predefined Values: None

RFC: RFC 4538

#### 11.2.2. remote-tag

Header Field: Target-Dialog

Header Field Parameter: remote-tag

Predefined Values: None

RFC: RFC 4538

### 11.3. 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.

## 12. Acknowledgements

This specification is based on a header field first proposed by Robert Sparks in the dialog usage draft [12]. John Elwell provided helpful comments.

## 13. References

### 13.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [3] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.
- [4] Donovan, S., "The SIP INFO Method", RFC 2976, October 2000.
- [5] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.
- [6] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.
- [7] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.
- [8] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

- [9] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, October 2004.
- [10] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", BCP 98, RFC 3968, December 2004.

### 13.2. Informative References

- [11] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [12] Sparks, R., "Multiple Dialog Usages in the Session Initiation Protocol", Work in Progress, March 2006.
- [13] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", Work in Progress, May 2006.
- [14] Rosenberg, J., "A Framework for Application Interaction in the Session Initiation Protocol (SIP)", Work in Progress, July 2005.
- [15] Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005.
- [16] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Conference State", Work in Progress, July 2005.
- [17] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.
- [18] Burger, E., "A Session Initiation Protocol (SIP) Event Package for Key Press Stimulus (KPML)", Work in Progress, December 2004.
- [19] Sparks, R., Ed., Hawrylyshen, A., Johnston, A., Rosenberg, J., and H. Schulzrinne, "Session Initiation Protocol (SIP) Torture Test Messages", RFC 4475, May 2006.

## Author's Address

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

Phone: +1 973 952-5000  
EMail: [jdrosen@cisco.com](mailto:jdrosen@cisco.com)  
URI: <http://www.jdrosen.net>



## Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

