SIP WG R. Mahy
Internet-Draft Cisco Systems, Inc.
Expires: December 29, 2002 D. Petrie
Pingtel
June 30, 2002

The Session Inititation Protocol (SIP) "Join" Header

draft-mahy-sip-join-and-fork-01.txt

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on December 29, 2002.

Copyright Notice

Abstract

This document defines a new header for use with SIP multi-party applications and call control. The Join header is used to logically join an existing SIP dialog with a new SIP dialog. This primitive can be used to enable a variety of features, for example: "Barge-In", answering-machine-style "Message Screening" and "Call Center Monitoring". Note that definition of these example features is non-normative.

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].

This document refers frequently to the terms "confirmed dialog" and "early dialog". These are defined in Section 12 of SIP [1].

2. Overview

This document describes a SIP [1] extension header field as part of the SIP multiparty applications architecture framework [8]. The Join header is used to logically join an existing SIP dialog with a new SIP dialog. This is especially useful in peer-to-peer call control environments.

One use of the "Join" header is to insert a new participant into a multimedia conversation (which may a two-party call or a conference). While this functionality is already available using 3rd party call control [12] style call control, the 3pcc model requires a central point of control which may not be desirable in many environments. As such, a method of performing these same call control primitives in a distributed, peer-to-peer fashion is very desirable.

Use of an explicit Join header is needed in some cases instead of addressing an INVITE to a conference URI for the following reasons:

- o A conference may not exist--the new invitation may be trying to join an oridinary two-party call.
- o The party joining may not know if the dialog it wants to join is part of a conference.
- o The party joining may not know the conference URI.

The Join header enables services such as barge-in, real-time message screening, and call center monitoring in a distributed peer-to-peer way. This list of services is not exhaustive.

For example, the Boss has an established 2-party conversation with a Customer, and using some out-of-band mechanism (ex:voice, gestures, and email) asks an Assistant to join the conversation. The Assistant sends an INVITE with a Join header to the Boss with the dialog information for the established dialog. The Assistant obtained this information from some mechanism, for example a web-page, and instant message, or from the SIP dialog package [9].

```
   Assitant     Boss        Customer
   | callid: 4@A |  callid: 7@c |
   |             |              |
   |             |<============>|
   |             |              |
   |INVITE------>|              |
   |Join: 7@c    |              |
   |             |reINVITE----->|
   |<----200-----|<----200------|
   |-----ACK---->|<----ACK------|
   |             |              |
   |   .. begins mixing ..      |
   |             |              |
   |<===========>|<============>|
   |<::::::::::::::::::::::::::>|
```

Note that this operation effectively creates a new conference. The Boss needs to start a new conference and create or obtain a new conference URI. In our example, the Boss mixes all media locally, so it needs to generate a new conference URI, return the conference URI as the Contact to the Join INVITE, and reINVITE the Customer with the conference URI as the new Contact.

3. Applicability of RFC2804 ("Raven")

This primitive can be used to create services which are used for monitoring purposes, however these services do not meet the definition of a wiretap according to RFC2804 [10]. The definition from RFC2804 is included here:

> Wiretapping is what occurs when information passed across the Internet from one party to one or more other parties is delivered to a third party:
>
> 1. Without the sending party knowing about the third party
> 2. Without any of the recipient parties knowing about the delivery to the third party
> 3. When the normal expectation of the sender is that the transmitted information will only be seen by the recipient parties or parties obliged to keep the information in confidence
> 4. When the third party acts deliberately to target the transmission of the first party, either because he is of interest, or because the second party's reception is of interest.

Specifically, item 2 of this definition does not apply to this extension, as one party is always aware of a Join request and can even decline such requests. In addition, in many applications of this primitive, some or all of the other items may not apply. For example, in many call centers which handle financial transactions, all conversations are recorded with the full knowledge and expectation of all parties involved.

4. User Agent Server Behavior: Receiving a Join Header

The Join header contains information used to match an existing SIP dialog (call-id, to-tag, and from-tag). Upon receiving an INVITE with a Join header, the UA attempts to match this information with a confirmed or early dialog. The to-tag and from-tag are matched as if they were present in an incoming request. In other words the to-tag is compared to the local tag, and the from-tag is compared to the remote tag.

If more than one Join header field is present in an INVITE, or if a Join header field is present in a request other than INVITE, the UAS MUST reject the request with a 400 Bad Request response.

The Join header has specific call control semantics. If both a Join header field and another header field with contradictory semantics (for example a Replaces [5] header field) are present in a request, the request MUST be rejected with a 400 "Bad Request" response.

If the Join header field matches more than one dialog, the UA MUST act as if no match is found.

If no match is found, the UAS rejects the INVITE and returns a 481 Call/Transaction Does Not Exist response. Likewise, if the Join header field matches a dialog which was not created with an INVITE, the UAS MUST reject the request with an appropriate response (ex: 400, 481, or 501).

If the Join header field matches a dialog which has already terminated, the UA SHOULD decline the request with a 603 Declined response.

If the Join header field matches a active dialog, the UA SHOULD verify that the initiator of the new INVITE is authorized to join the matched dialog. If the initiator of the new INVITE has authenticated successfully as equivalent to the user who is being joined, then the join is authorized. The UA MAY maintain a list of authorized entities who are allowed to join any dialog with certain characteristics (for example, all dialogs placed in the call center context of the UA). In addition, the UA MAY use other authorization mechanisms defined for this purpose in standards track extensions. For example, an extension could define a mechanism for transitively asserting authorization of a join.

If authorization is successful, the UA attempts to accept the new INVITE, and assign any mixing or conferencing resources necessary to complete the join. If the UA cannot accept the new INVITE (for example: it cannot establish required QoS or keying, or it has incompatible media), the UA MUST return an appropriate error response and MUST leave the matched dialog unchanged.

A User Agent that accepts a Join header needs to setup dialogs or conferences such that the requesting UAC is logically added to the conversation space associated with the matched dialog. Any dialogs which are already logically associated with the matched dialog in the same conversation space are included as well. All the participants in a conversation space should have access to all the media/content sent in the context of that conversation space. That a participant does not negotiate a specific type of media does not mean that it is not otherwise a full participant. For a detailed description of various conferencing mechanisms that could be used to handle a Join, please consult the SIP conferencing framework [6].

If the UAS has sufficient resources to locally handle the Join request, the UAS SHOULD accept the Join request and perform the appropriate media mixing or combining. The UAS MAY rearrange appropriate dialogs instead as described below, based on some local policy.

If the UAS does not have sufficient resources locally to handle the request, or does not wish to use these local resources, but is aware of other resources which could be used to satisfy the request (ex: a centralized mixer), the UA SHOULD create a conference using this resource (ex: INVITE the centralized mixer to obtain a conference URI), redirect the requestor to this resource, and request other participants in the same conversation space to use this resource. The UA MAY use any appropriate mechanism to transition participants to the new resource (ex: 3xx repsonse, 3rd-party call control reinvitiations, REFER requests, or reinvitations to a multicast group). The UA SHOULD only use mechanisms which are expected to be acceptable to the participants. For example, the UA SHOULD NOT attempt to transition the participants to a multicast group unless the UA can reasonably expect that all the particpants can support multicast.

If the UAS is incapable of satisfying the Join request, it MUST return a 488 "Not Acceptable Here" response.

5. User Agent Client Behavior: Sending a Join header

A User Agent that wishes to add a new dialog of its own to a single existing early or confirmed dialog and any associated dialogs or conferences, MAY send the target User Agent an INVITE request containing a Join header field. The UAC places the Call-ID, to-tag, and from-tag information for the target dialog in a single Join header field and sends the new INVITE to the target.

Note that use of this mechanism does not provide a way to match multiple dialogs, nor does it provide a way to match an entire call, an entire transaction, or to follow a chain of proxy forking logic. For example, if Alice replaces Cathy in an early dialog with Bob, but he does not answer, Alice's replacement request will not match other dialogs to which Bob's UA redirects, nor other branches to which his proxy forwards.

6. Proxy behavior

Proxy Servers do not require any new behavior to support this extension. They simply pass the Join header field transparently as described in the SIP specification.

Note that it is possible for a proxy (especially when forking based on some application layer logic, such as caller screening or time-of-day routing) to forward an INVITE request containing a Join header field to a completely orthogonal set of Contacts than the original request it was intended to replace. In this case, the INVITE request with the Join header field will fail.

7. Syntax

7.1 The Join Header

The Join header field indicates that a new dialog (created by the INVITE in which the Join header field in contained) should be joined with a dialog identified by the header field, and any associated dialogs or conferences. It is a request header only, and defined only for INVITE requests. The Join header field MAY be encrypted as part of end-to-end encryption. Only a single Join header field value may be present in a SIP request

This document adds the following entry to Table 3 of [1]. Additions to this table are also provided for extension methods defined at the time of publication of this document. This is provided as a courtesy to the reader and is not normative in any way. SUBSCRIBE and NOTIFY, REFER, INFO, UPDATE, and PRACK are defined respectively in [14], [4], [15], [16], and [17].

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Join	R		-	-	-	o	-	-
			SUB	NOT	REF	INF	UPD	PRA
			---	---	---	---	---	---
Join	R		-	-	-	-	-	-

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [3].

```
Join            = "Join" HCOLON callid *(SEMI join-param)
join-param      = to-tag / from-tag / generic-param
to-tag          = "to-tag" EQUAL token
from-tag        = "from-tag" EQUAL token
```

A Join header MUST contain exactly one to-tag and exactly one from-tag, as they are required for unique dialog matching. For compatibility with dialogs initiated by RFC2543 [7] compliant UAs, a tag of zero matches both tags of zero and null tags.

Examples:

```
Join: 98732@sip.billybiggs.com
       ;from-tag=r33th4x0r
       ;to-tag=ff87ff
Join: 12adf2f34456gs5;to-tag=12345;from-tag=54321
Join: 87134@171.161.34.23;to-tag=24796;from-tag=0
```

7.2 New option tag for Require and Supported headers

This specification defines a new Require/Supported header option tag "join". UAs which support the Join header MUST include the "join" option tag in a Supported header field. UAs that want explicit failure notification if Join is not supported MAY include the "join" option in a Require header field.

Example:

```
Require: join, 100rel
```

8. Usage Examples

The following non-normative examples are not intended to enumerate all the possibilities for the usage of this extension, but rather to provide examples or ideas only. For more examples, please see service-examples [13].

8.1 Join accepted and transitioned to central mixer

```
A             B              C             mixer
| callid: 4@A |  callid: 7@c |              |
|             |              |              |
|             |<============>|              |
|             |              |              |
|INVITE------>|              |              |
|Join: 7@c    |--INVITE--------------------->|
|             |<----200----------------------|
|             |-----ACK--------------------->|
|<----300-----|                             |
|INVITE------------------------------------>|
|<--200-------------------------------------|
|---ACK------------------------------------>|
|             |--REFER------>|              |
|             |<---200-------|--INVITE----->|
|             |              |<----200------|
|             |<--NOTIFY-----|-----ACK----->|
|             |------200---->|              |
|             |---BYE------->|              |
|             |<--200--------|              |
|             |              |              |
|<========================================>| mixes the
|             |<============================>| three sessions
|             |              |<============>| together
```

The conversation space now looks identical to the locally mixed example in the Introduction. Details of how the Join are implemented are transparent to A. B could have used 3rd party call control instead to move the necessary sessions.

```
[ B , C ]  -->  [ A , B , C ]
```

8.2 Join rejected

```
   A             B              C
   | callid: 4@A |  callid: 7@c |
   |             |              |
   |             |<============>|
   |             |              |
   |INVITE------>|              |
   |Join: 7@c    |              |
   |             |              |
   |<----486-----|              |
   |-----ACK---->|              |
   |             |              |
```

In this example B is Busy (does not want to be disturbed), and therefore does not wish to add A. B could also decline the request with a 603 response.

9. Security Considerations

The extension specified in this document significantly changes the relative security of SIP devices. Currently in SIP, even if an eavesdropper learns the Call-ID, To, and From headers of a dialog, they cannot easily modify or destroy that dialog if Digest authentication or end-to-end message integrity are used.

This extension can be used to insert or monitor potentially sensitive content in a multimedia conversation. As such, invitations with the Replaces header SHOULD only be accepted if the peer requesting replacement has been properly authenticated using a standard SIP mechanism, and authorized to by joined with the target dialog.

Some mechanisms for obtaining the dialog information needed by the Join header (Call-ID, to-tag, and from-tag) include URIs on a web page, subscriptions to an appropriate event package, and notifcations after a REFER request. Use of end-to-end security mechanisms to encrypt this information is also RECOMMENDED.

This extension was designed to take advantage of future signature or authorization schemes defined by the SIP Working Group. In general, call control features would benefit considerably from such work.

10. IANA Considerations

10.1 Registration of "Join" SIP header

Name of Header: Join

Short form: none

Normative description: section 7.1 of this document

10.2 Registration of "join" SIP Option-tag

Name of option: join

Description: Support for the SIP Join header

SIP headers defined: Join

Normative description: This document

11. Changes

11.1 Changes Since -00

- **Realigned the text to mirror the outline of Replaces**
- Removed the fork header
- Added a section to explain how this is not a "Raven" wiretap mechanism
- Reorganized motivational overview material
- Added authorization language in UAS behavior section
- Updated and Added references

12. Acknowledgments

Thanks to Robert Sparks, Alan Johnston, and Ben Campbell and many other members of the SIP WG for their continued support of the cause of distributed call control in SIP.

Normative References

[1] Rosenberg, J. and H. Schulzrinne, "SIP: Session Initiation Protocol", draft-ietf-sip-rfc2543bis-09 (work in progress),

February 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.

Informational References

[4] Sparks, R., "The SIP Refer Method", draft-ietf-sip-refer-05 (work in progress), June 2002.

[5] Dean, R., Biggs, B. and R. Mahy, "The Session Inititation Protocol (SIP) 'Replaces' Header", draft-ietf-sip-replaces-02 (work in progress), May 2002.

[6] Rosenberg, J. and H. Schulzrinne, "Models for Multi Party Conferencing in SIP", draft-ietf-sipping-conferencing-models-00 (work in progress), November 2001.

[7] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[8] Mahy, R., "A Multi-party Application Framework for SIP", draft-ietf-sipping-cc-framework-00 (work in progress), March 2002.

[9] Rosenberg, J. and H. Schulzrinne, "A Session Initiation Protocol (SIP) Event Package for Dialog State", draft-ietf-sipping-dialog-package-00 (work in progress), June 2002.

[10] IESG, IAB,., "IETF Policy on Wiretapping", RFC 2804, May 2000.

[11] Sparks, R., "SIP Call Control - Transfer", draft-ietf-sip-cc-transfer-05.txt (work in progress), July 2001.

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

[13] Johnston, A., "SIP Service Examples", draft-ietf-sipping-service-examples-01 (work in progress), April 2002.

[14] Roach, A., "SIP-Specific Event Notification", draft-ietf-sip-events-05 (work in progress), March 2002.

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

[16] Rosenberg, J., "The Session Initiation Protocol UPDATE Method", draft-ietf-sip-update-02 (work in progress), May 2002.

[17] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in SIP", draft-ietf-sip-100rel-06 (work in progress), February 2002.

Authors' Addresses

Rohan Mahy
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA
EMail: rohan@cisco.com

Dan Petrie
Pingtel
400 West Cummings Park, Suite 2200
Woburn, MA 01801
USA
EMail: dpetrie@pingtel.com

Acknowledgement

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