Abstract

The current specification for the SIP MESSAGE request method indicates that SIP instant messages according to a model similar to that of a text pager, in that each message stands alone. There is no concept of a chat session or a text conference where there is a stream of messages that are grouped into a session. This memo proposes a method of describing MESSAGE sessions by treating the message session just like any other media session described in an SDP body in an INVITE request.
Table of Contents

1. Introduction .................................................. 3
2. Motivation ...................................................... 3
3. Overview of Operation .......................................... 4
4. User Agent Client Behavior ................................. 5
5. User Agent Server Behavior ................................. 6
6. Nature of MESSAGE sessions ................................. 6
7. Ending Message Sessions ..................................... 7
8. Identifying Sessions ......................................... 7
9. Message Sessions over SIP Proxies ....................... 7
10. Security Considerations ...................................... 8
11. Open Issues .................................................. 9
12. Acknowledgments ............................................. 10
    References .................................................. 10
    Authors’ Addresses ........................................ 10
    Full Copyright Statement .................................. 12
1. Introduction

The SIP MESSAGE method does not currently support any sense of a session. Instant messages sent using this method are treated like pager messages. Each message stands alone, and is not linked into a conversation. There has been recent interest in the idea of a SIP based instant message session, where the user experience is more akin to a text conference or a chat room. This document proposes the idea of treating SIP instant message sessions as a media type that can be initiated using the same SIP mechanisms as for any other media type.

In this approach, a SIP endpoint that wishes to initiate a text chat session would send an INVITE request with an SDP body that describes the session [2]. The sender and recipient then negotiate MESSAGE sessions using normal SIP conventions.

2. Motivation

The SIP MESSAGE request method [3] does not create a session. Each MESSAGE request/response exchange fully stands alone, and is not affected by previous exchanges. This is a perfectly good model for many uses of instant messages, such as SMS messages to wireless devices, etc. This document in no way deprecates the stand-alone model of instant messaging, as a session concept is an undue burden for a single message. However, many Instant Message applications support the concept of a message session in the form of a conference or a chat room, in which two (or commonly more) users hold a conversation that is displayed as a coherent whole.

The stand-alone model has a number of limitations. It only supports multi-party messaging in very clumsy ways, while using INVITE to establish a session allows reuse of the various multi-party conferencing models supported by SIP[5].

The stand-alone model by necessity causes MESSAGE requests to follow the SIP signal path, which is intended to manage sessions, not transfer content.

Forking of MESSAGE requests does not work well either. The sender will not know how many recipients have gotten the message. The sender will not be able to select one of those recipients as the target for future messages. These problems are resolved by establishing a session with INVITE. In that case, the caller does know who has received the session invitation, and can select which recipient(s) to communicate with.

Finally, MESSAGE sessions treated as a normal media stream allow us to combine MESSAGE streams with other types of media. For example,
one could establish a conference call with a text sub-channel, or send closed captioning along with a video stream.

3. Overview of Operation

Let us imagine a SIP endpoint Alice, which wishes to initiate a chat session with Bob. Alice constructs an INVITE request with an SDP body, and includes the following line in the SDP:

\[
\text{m=message 5060 sip sip:alice@alicepc}
\]

Bob’s UAS receives the INVITE, and responds with a 200 OK containing the following SDP line:

\[
\text{m=message 5061 sip sip:bob@bobpda:5061}
\]

Finally, Alice follows up with an ACK request.

At this point, Alice and Bob can each send MESSAGE requests directly to the other. Note that each direction is an independent media stream, and will have a distinct combination of To, From, and Call-ID headers, as well as distinct CSeq spaces.

Additionally, the To, From, and Call-ID headers and CSeq spaces are independent from those of the signaling session.

The following call flow illustrates the basic message session model, where the signal path goes through a record-routing proxy, but the message session path is end-to-end.
4. User Agent Client Behavior

A UAC wishing to initiate a MESSAGE session MUST construct and
INVITE request. The headers of the INVITE must be generated
according to the normal rules of SIP [1]. The methods for
negotiating the MESSAGE streams is the same as for SIP in general,
except that the UAC MUST include an SDP body in the INVITE
containing the description of the desired inbound MESSAGE session,
i.e. the URL at which it would like to receive MESSAGE requests as
part of this session. The SDP format for describing SIP message
sessions is described in the SIP Message SDP draft [2] This is
different than in general SIP which allows the UAC to defer
proposing its media selection until the ACK request.

It is tempting to allow the UAC to defer the media specification,
so that the SDP is carried in the 200 class response and the ACK
request, instead of in the INVITE request and the 200 class
response. This would be useful for 3PCC style applications.
However, if the UAC omits the media description for the MESSAGE
session, there is a high likelihood that the UAS will not propose
a message media type in the 200 response. There is still a
possibility for the UAC to accept a non-text media type proposed
by the UAS, but it will no longer be possible to accomplish the
original goal, i.e. establish a MESSAGE session.

The UAC MUST be prepared to receive MESSAGE requests to its proposed
URL prior to the completion of the INVITE transaction; that is,
before receiving a 200 class response or sending an ACK request.
5. User Agent Server Behavior

A UAS that receives an INVITE containing an SDP description handles media negotiation according to the normal rules of SIP[1]. Since the message media type does not have a concept of codecs, negotiation of message media sections is considerably more simple than for RTP media sections.

The UAS must be prepared to receive MESSAGE request to its proposed URL prior to the completion of the INVITE transaction; that is, before receiving an ACK request.

6. Nature of MESSAGE sessions

A MESSAGE session takes the form of a SIP call-leg, and is identified in the same way as general SIP call legs. A message session is peculiar in the fact that each call-leg is one way only. A two-way chat between two SIP endpoints contains two call legs; one in each direction. Each endpoint may send MESSAGE requests to the URL that was advertised in the opposite end-points SDP. An endpoint MUST NOT send any request other than MESSAGE on a message session call leg, and it MUST NOT violate the directionality of the call-leg; that is to say it SHOULD respond to MESSAGE requests sent to it, but it MUST NOT attempt to send MESSAGE request back along the same call leg. If an endpoint receives a request that violates directionality, it SHOULD respond with a 481, as if the call leg never existed. If an endpoint receives a request with a method other than MESSAGE it SHOULD respond with a 405.

When a UAC sends a MESSAGE request on a session, it MUST set the Request-URI to the URL specified in its opposite’s SDP, and send the request according to normal SIP rules for resolving a Request-URI. If the supplied URL contains headers, the UAC MUST include those headers in its MESSAGE request. One exception is CSeq; if the URL included a CSeq header, the UAC SHOULD ignore it, and generate its initial CSeq normally.

The UAC MUST repeat this process for each MESSAGE requests it sends, following normal SIP rules for sending requests on a call-leg. Note that since MESSAGE requests and their responses will not contain CONTACT headers, each MESSAGE request on a call leg MUST be sent along the same path as the initial request. A MESSAGE request MUST include a pre-loaded route set if the advertised URL contained ROUTE headers.

The MESSAGE method is specified to have no session semantics into itself. In particular, a proxy should not insert a Record-Route header into a MESSAGE request[3]. To do would have no meaning.
Even though the MESSAGE request may occur in the context of a message session, they may go through a proxy that has no knowledge of the session, and will treat each request as a stand-alone MESSAGE. If that proxy needs to stay in the MESSAGE path for one reason or another (for example, it is a firewall proxy) it cannot use Record-Route to accomplish this. If subsequent MESSAGE requests went to the URL in a Contact header, it would not be possible for the proxy to stay in the path.

MESSAGE requests inside a message session MUST NOT overlap. That is, when the UAC sends one MESSAGE request, it may not send another until the first transaction has completed.

7. Ending Message Sessions

Under normal conditions, SIP endpoints use the BYE request on the signal channel to end a message session just like they would with any other session. If the session is bi-directional, that is, consists of two unidirectional session call legs, the BYE request tears down both call-legs. And endpoint MUST not attempt to send the bye in the message session itself.

If a UACs attempt to send a MESSAGE request fails, either do to a negative response from the UAS or a timeout waiting for a response, it should behave as it would for any other media stream failure.

8. Identifying Sessions

A SIP endpoint MUST advertise a URL which allows it to determine which call leg an incoming message is for. It can do that by embedding a Call-ID header in the URL, or by using a special user name unique to the call leg.

9. Message Sessions over SIP Proxies

In a perfect world, all message sessions would be end-to-end. In this case, end-to-end refers to the scenario where the URL advertised by each endpoint directly maps to the endpoint itself. In reality, there are situations where message sessions may need to cross SIP proxies. For example, an endpoint may wish to conceal its address, or may be behind a firewall where all traffic must go through a particular proxy.

In the simplest proxy case, the endpoint merely advertises a URL to a proxy that knows how to route MESSAGE requests to the desired endpoints. For example, a the endpoint might send the following registration:
Then the endpoint could send an invite where the SDP contained sip:bob@biloxi.com. In this case, all MESSAGE requests sent on the session would transverse the proxy.

Another approach is to place a pre-loaded route header directly to the advertised URL. For example, Alice might include the following m-line in the body of the INVITE she sent to Bob:

```
m=message 5060 sip
    sip:alice@atlanta.com?Route=sip:alicepc.atlanta.com&Call-ID=34reid2jk@alicepc.atlanta.com
```

Bob then sends each MESSAGE request to the proxy at sip:atlanta.com with a pre-loaded route header instructing the proxy to route the request to sip:alicepc.atlanta.com.

And endpoint may also need to route MESSAGE requests through a default outbound proxy, regardless of whether they are in-session or stand-alone. Since each request follows the same path as the initial request this will work without a problem. It is interesting to note that the default outbound proxy will stay in the message session path even if it does not request Record-Routing.

10. Security Considerations

Message sessions have all of the same security considerations as any other media type used with SIP sessions. A primary issue is the exposure of end-point addresses to the opposite end-point (and anything in between.) This is slightly mitigated with message sessions, as an intervening proxy may be used to hide the end-point address. The deprecation of Contact headers in MESSAGE transactions also mitigates that issue slightly.

Another issue is the possibility that a rogue endpoint could establish a message session in the absence of an INVITE transaction by simply guessing the URL to which to send messages. This opportunity is reduced if the endpoints add difficult to guess Call-ID headers into the URLs advertised in their SDP, and only accept MESSAGES with a matching call ID. Still, an attacker could
sniff the network and capture the Call-ID if the INVITE transaction is transmitted in the clear. Note that this problem is much worse for audio media streams since they only have 64k ports to guess from, while Call-ID space is effectively infinite.

An endpoint may apply any of the general SIP authentication methods to message sessions.

### 11. Open Issues

We assert that all requests in a message session must follow the same path as the initial MESSAGE request. But what if we encounter a redirect server? It seems inefficient to require each request to individually get redirected, instead of remembering the contact from the redirect server.

The inability for a proxy to Record-Route a MESSAGE request causes some ugliness. Since we are then forced to send all request via the same path as the original MESSAGE, we make it impossible for a proxy to _not_ stay in the message session path. At the same point in time, it would be really ugly to have different semantics for MESSAGE requests depending on whether they were in-session or stand-alone.

Another point that came up on the list is we need a way to tell and endpoint to route messages along the signal path. This draft does not address that problem. It might be possible to simply leave the URL out of the m-line, and have endpoints recognize that to mean to use the signal path. I am not confident enough in that approach to propose it in the main body of this draft.

Does this draft need to discuss negotiating what mime-types are permitted in the bodies of in-session MESSAGE requests?

This approach handles forking of the INVITE transaction just fine. But what if the message session itself crosses a forking proxy? Each subsequent request in the call leg will also fork. Is this a problem?

Robert brought up a concern about the performance of message sessions, since we cannot overlap MESSAGE transactions in a call-leg. The fact that each call-leg is one-way helps this a little, but it is still sub-optimal.

Do we need to make special considerations for message sessions over TLS?

We have effectively introduced the idea of SIP call leg nesting. Should this be generalized?
12. Acknowledgments

This document is a compilation of the thoughts of many people on the SIMPLE mail list. In particular, the authors thank the following for their contributions: Christian Heitema, Sean Olson, Adam Roach, Robert Sparks, Henning Schulzrinne, and James Undery.

References


Authors’ Addresses

Ben Campbell
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX  75024

e-mail: bcampbell@dynamicsoft.com
Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936

e-mail: jdrosen@dynamicsoft.com