Session Initiation Protocol Extension for Instant Messaging
draft-ietf-sip-message-04

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 November 1, 2002.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

Instant Messaging (IM) refers to the transfer of messages between users in near real-time. These messages are usually, but not required to be, short. IMs are often used in a conversational mode, that is, the transfer of messages back and forth is fast enough for participants to maintain an interactive conversation.
The MESSAGE method is an extension to the Session Initiation Protocol (SIP) that allows the transfer of Instant Messages. MESSAGE requests carry the content in the form of MIME body parts. MESSAGE requests do not themselves initiate a SIP dialog; under normal usage each Instant Message stands alone, much like pager messages. MESSAGE requests may be sent in the context of a dialog initiated by some other SIP request.

Since the MESSAGE request is an extension to SIP it inherits all the request routing and security features of that protocol.

Table of Contents

1. Introduction ........................................... 4
2. Scope of Applicability .................................. 4
3. Overview of Operation ................................. 5
4. UAC Processing ........................................ 6
5. Use of Instant Message URIs .......................... 7
6. Proxy Processing ...................................... 7
7. UAS Processing ......................................... 8
8. Caller Preferences ..................................... 8
9. Congestion Control ..................................... 9
10. Method Definition .................................... 10
11. Example Messages ..................................... 12
12. Security Considerations .............................. 13
12.1 Outbound authentication ............................. 14
12.2 SIPS URIs ........................................... 14
12.3 End-to-End Protection ............................... 14
12.4 Replay Prevention ................................... 14
12.5 Using message/cpim bodies ......................... 15
13. IANA Considerations .................................. 15
14. Changes to This Document ............................ 15
14.1 Changes introduced in draft-ietf-sip-message-04 ...... 15
14.2 Changes introduced in draft-ietf-sip-message-03 ...... 16
14.3 Changes introduced in draft-ietf-sip-message-02 ...... 16
14.4 Changes introduced in draft-ietf-sip-message-01 ...... 17
14.5 Changed Introduced in draft-ietf-sip-message-00 ...... 17
14.6 Changes Introduced in draft-ietf-simple-im-01 ....... 17
14.7 Changes Introduced in draft-ietf-simple-im-00 ....... 17
14.8 Changes Introduced in draft-rosenberg-impp-im-01 .... 17
15. Contributors .......................................... 18
16. Acknowledgments ...................................... 18
      Normative References ................................. 18
      Informational References ............................ 19
      Authors’ Addresses .................................. 19
      Full Copyright Statement ............................ 21
**1. Introduction**

Instant messaging is defined as the exchange of content between a set of participants in near real time. Generally, the content is short text messages, although that need not be the case. Generally, the messages that are exchanged are not stored, but this also need not be the case. IM differs from email in common usage in that instant messages are usually grouped together into brief live conversations, consisting of numerous small messages sent back and forth.

Instant messaging as a service has been in existence within intranets and IP networks for quite some time. Early implementations include zephyr [8], the UNIX talk application, and IRC. More recently, IM has been used as a service coupled with presence and buddy lists; that is, when a friend comes online, a user can be made aware of this and have the option of sending the friend an instant message. The protocols for accomplishing this are all proprietary, which has seriously hampered interoperability.

The integration of instant messaging, presence, and session-oriented communications is very powerful. The Session Initiation Protocol (SIP) [1]provides mechanisms that are useful for presence applications, and for session-oriented communication applications, but not for instant messages.

This document proposes an extension method for SIP called the MESSAGE method. MESSAGE requests normally carry the instant message content in the request body.

RFC2778 [7]and RFC2779 [6]give a model and requirements for presence and instant messaging protocols. The MESSAGE method is intended to meet the instant messaging requirements therein.

**2. Scope of Applicability**

This document describes the use of the MESSAGE method for sending instant message using a metaphor similar to that of a two-way pager or SMS enabled handset. That is, there are no explicit association between messages. Each IM stands alone—any sense of a "conversation" only exists in the client user interface, or perhaps in the user’s own imagination. We contrast this with a "session" model, where there is an explicit conversation with a clear beginning and end. In the SIP environment, an IM session would be a media session initiated with an INVITE transaction and terminated with a BYE transaction.

There is value in each model. Most modern IM clients offer both user experiences. The user can choose to send an IM to a contact, or he
can choose to invite one or more contacts to join a conversation. The pager model makes sense when the user wishes to send a small number of short IMs to a single (or small number of) recipients. The session model makes sense for extended conversations, joining chat groups, there is a need to associate a conversation with some other SIP initiated session, etc.

This document addresses the pager model only. We recognize the value of the session model as well; but we do not define it here. Such a solution will require additional work beyond that of this document. The SIMPLE work group currently plans to address IM sessions in a separate document.

There may be a temptation to simulate a session of IMs by initiating a dialog, then sending MESSAGE requests in the context of that dialog. This is not an adequate solution for IM sessions, in that this approach forces the MESSAGE requests to follow the same network path as any other SIP requests, even though the MESSAGE requests arguably carry media rather than signaling. IM applications are typically high volume, and we expect the IM volume in sessions to be even higher. This will likely cause congestion problems if sent over a transport without congestion control, and there is no clear mechanism in SIP to prevent some hop from forwarding a MESSAGE request over UDP.

Additionally, MESSAGE requests sent over an existing dialog must, by the nature of SIP, go to the same destination as any other request sent in that dialog. This prevents any separation between the IM endpoint and the signaling endpoint. This is not an acceptable limitation for the session-model of instant messaging.

The author’s recognize that there may be valid reasons to send MESSAGE requests in the context of a dialog. For example, one participant in a voice session may wish to send an IM to another participant, and associate that IM with the session. But implementations MUST NOT create dialogs for the primary purpose of associating MESSAGE requests with one another.

Note that this statement does not prohibit using SIP to initiate a media session made up of IMs, just like any other session. Indeed, we expect the solution for IM sessions to use that metaphor. The reader should avoid confusing the concepts of a SIP dialog and a media session.

3. Overview of Operation

When one user wishes to send an instant message to another, the sender formulates and issues a SIP request using the new MESSAGE

Campbell (Editor), et al. Expires November 1, 2002
method defined by this document. The request URI of this request will normally be the "address of record" for the recipient of the instant message, but it may be a device address in situations where the client has current information about the recipient’s location. For example, the client could be coupled with a presence system that supplies an up to date device contact for a given address of record. The body of the request will contain the message to be delivered. This body can be of any MIME type, including message/cpim. [4]

The request may traverse a set of SIP proxies, using a variety of transports, before reaching its destination. The destination for each hop is located using the address resolution rules detailed in the Common Profile for Instant Messaging (CPIM) [3] and SIP specifications. During traversal, each proxy may rewrite the request URI based on available routing information.

Provisional and final responses to the request will be returned to the sender as with any other SIP request. Normally, a 200 OK response will be generated by the user agent of the request’s final recipient. Note that this indicates that the user agent accepted the message, not that the user has seen it.

MESSAGE requests do not establish dialogs.

4. UAC Processing

Unless stated otherwise in this document, MESSAGE requests and associated responses are constructed according to the rules in section 8.1 of the SIP specification. [1]

All UAs which support the MESSAGE method MUST be prepared to send and receive MESSAGE requests with a body of type text/plain. They MAY send bodies of type message/cpim.

MESSAGE requests do not initiate dialogs. User Agents MUST not insert contact headers into MESSAGE requests.

A UAC MAY associate a MESSAGE request with an existing dialog. If a MESSAGE request is sent within a dialog, it is "associated” with any media session or sessions associated with that dialog.

If the UAC receives a 200 OK response to a MESSAGE request, it may assume the message has been delivered to the final destination. It MUST NOT assume that the recipient has actually read the instant message. If the UAC receives a 202 Accepted response, the message has been delivered to a gateway, store and forward server, or some other service that may eventually deliver the message. In this case, the UAC MUST NOT assume the message has been delivered to the final
destination. If confirmation of delivery is required for a message that has been responded to with a 202 Accepted, that confirmation must be delivered via some other mechanism, which is beyond the scope of this specification.

Note that a downstream proxy could fork a MESSAGE request. If this occurs, the forking proxy will forward one final response upstream, even though it may receive multiple final responses. The UAC will have no way to detect whether or not a fork occurs. Therefore the UAC MUST NOT assume that a given final response represents the only UAS that receives the request. For example, multiple branches of a fork could have resulted in 2XX class responses. Even though the UAC only sees one of those responses, the request has in fact been received by the second device as well.

5. Use of Instant Message URIs

An instant inbox may be most generally referenced by an Instant Message URI [3] in the form of "im:user@domain". IM URIs are abstract, and MUST eventually be translated to concrete, protocol-dependent URI using the method described in the CPIM specification.

If a UA is presented with an IM URI as the address for an instant message, it SHOULD resolve it to a SIP URI, and place the resulting URI in the Request-URI of the MESSAGE request before sending. If the UA is unable to resolve the IM URI, it MAY place the IM URI in the Request-URI, thus delegating the resolution to a downstream device such as proxy or gateway. Performing this translation as early as possible allows SIP proxies, which may be unaware of the im: namespace, to route the requests normally.

MESSAGE requests also contain logical identifiers of the sender and intended recipient, in the form of the From and To headers. These identifiers SHOULD contain SIP (or SIPS) URIs, but MAY include IM URIs if the SIP URIs are not known at the time of request construction.

Record-Route and Route headers MUST NOT contain IM URIs. These headers contain concrete SIP or SIPS URIs according to the rules of SIP. [1]

6. Proxy Processing

Proxies route MESSAGE requests according to the rules of SIP [1] for proxy routing of requests that do not initiate dialogs. Note that the MESSAGE request MAY fork; this allows for delivery of the message to several possible terminals where the user might be. A proxy
forking a MESSAGE request follows the normal SIP rules for forking a non-invite request. In particular, even if the fork results in multiple successful deliveries, the forking proxy will only forward one final response upstream.

7. UAS Processing

A UAS that receives a MESSAGE request processes it following the rules of SIP. [1]

A UAS receiving a MESSAGE request SHOULD respond with a final response immediately. Note, however, that the UAS is not obliged to display the message to the user either before or after responding with a 200 OK. That is, a 200 OK response does not necessarily mean the user has read the message.

A 2XX class response to a MESSAGE request MUST NOT contain a body. A UAS MUST NOT insert a contact header into a 2XX class response.

A UAS which is, in fact, a message relay, storing the message and forwarding it later on, or forwarding it into a non-SIP domain, SHOULD return a 202 (Accepted) [5] response indicating that the message was accepted, but end to end delivery has not been guaranteed.

A 4XX or 5XX class response indicates that the message was not delivered successfully. A 6XX response means it was delivered successfully, but refused.

A UAS that supports the MESSAGE method MUST be prepared to receive and interpret body types of "text/plain" and "message/cpim". [4]

8. Caller Preferences

User agents SHOULD add the "methods" tag defined in the caller preference [2] specification to Contact headers with SIP URIs placed in REGISTER requests, indicating support for the MESSAGE method. Other elements of caller preferences MAY be supported. For example:

```
REGISTER sip:dynamicsoft.com SIP/2.0
Via: SIP/2.0/UDP mypc.dynamicsoft.com
To: sip:jdrosen@dynamicsoft.com
From: sip:jdrosen@dynamicsoft.com
Call-ID: asidhasd@1.2.3.4
CSeq: 39 REGISTER
Contact: sip:jdrosen@im-pc.dynamicsoft.com;methods="MESSAGE"
Content-Length: 0
```
Registrar/proxies which wish to offer IM service SHOULD implement the proxy processing defined in the caller preferences specification.

9. Congestion Control

Existing IM services have a history of very high volume usage. Additionally, MESSAGE requests differ from other sorts of SIP requests in that they carry media, in the form of IMs, as payload. Conventional SIP payloads carry signaling information about media, but not media itself. There is potential that when a SIP infrastructure is shared between call signaling and instant messaging, the IM traffic will interfere with call signaling traffic. Congestion control in general is an issue that should be addressed at the SIP specification level rather than for an individual method. But since the traffic patterns are likely to be different for MESSAGE than for most other methods, it makes sense to give MESSAGE special consideration.

Whenever possible, MESSAGE requests SHOULD be sent over transports that implement end-to-end congestion control, such as TCP or SCTP. However, SIP does not provide a mechanism to prevent a downstream hop from sending a request over UDP. Even the requirement to use TCP for requests over a certain size can be overridden by the receiver. Therefore use of a congestion-controlled transport by the UAC is not sufficient.

The payload for MESSAGE requests outside of a media session MUST NOT exceed 1300 bytes. Larger payloads may be sent as part of a media session, or using some type of content-indirection. Implementations MUST NOT circumvent this rule by fragmenting content across multiple pager-model MESSAGE requests.

A UAC MUST NOT initiate a new out-of-dialog MESSAGE transaction to a given URI if there is a previous out-of-dialog transaction pending for the same URI. Similarly, A UAC SHOULD NOT initiate overlapping MESSAGE transactions inside a dialog, and MUST NOT do so unless the route set for that dialog uses a congestion-controlled transport at every hop. UACs SHOULD NOT set the T1 timer value to less than 500 ms for MESSAGE transactions. UACs may use smaller T1 values if they know that the next hop latency warrants it.

It has been suggested that provisional responses should not be used for pager-model MESSAGE requests. However, this is not possible, as many proxy servers will not be aware of the MESSAGE method, and will treat MESSAGE requests using the standard non-invite transaction. Additionally, prohibiting provisional responses may in some cases increase the number of retries, and actually make congestion problems worse. Therefore MESSAGE requests SHOULD receive the same
provisional response treatment as any other non-INVITE method, as described in the SIP specification.

10. Method Definition

This specification defines a new SIP method, MESSAGE. The BNF for this method is:

MESSAGEm = %x4D.45.53.53.41.47.45 ;MESSAGE in caps

As with all other methods, the MESSAGE method name is case sensitive.

Tables 1 and 2 extend Tables 2 and 3 of SIP [1] by adding an additional column, defining the headers that can be used in MESSAGE requests and responses.

<table>
<thead>
<tr>
<th>Header Field</th>
<th>where</th>
<th>proxy</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accept</td>
<td>2xx</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accept</td>
<td>415</td>
<td>m*</td>
<td></td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>2xx</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>415</td>
<td>m*</td>
<td></td>
</tr>
<tr>
<td>Accept-Language</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>2xx</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>415</td>
<td>m*</td>
<td></td>
</tr>
<tr>
<td>Alert-Info</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Alert-Info</td>
<td>180</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Allow</td>
<td>R</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Allow</td>
<td>2xx</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Allow</td>
<td>r</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Allow</td>
<td>405</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>Authentication-Info</td>
<td>2xx</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Authorization</td>
<td>R</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Call-ID</td>
<td>c</td>
<td>r</td>
<td>m</td>
</tr>
<tr>
<td>Call-Info</td>
<td>ar</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Contact</td>
<td>1xx</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Contact</td>
<td>2xx</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Contact</td>
<td>3xx</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td>485</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>Content-Disposition</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content-Encoding</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content-Language</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content-Length</td>
<td>ar</td>
<td>t</td>
<td></td>
</tr>
</tbody>
</table>
Table 1: Summary of header fields, A--O

<table>
<thead>
<tr>
<th>Header Field</th>
<th>where</th>
<th>proxy</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>R</td>
<td>ar</td>
<td>o</td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td>407</td>
<td>ar</td>
<td>m</td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td>401</td>
<td>ar</td>
<td>o</td>
</tr>
<tr>
<td>Proxy-Authorization</td>
<td>R</td>
<td>dr</td>
<td>o</td>
</tr>
<tr>
<td>Proxy-Require</td>
<td>R</td>
<td>ar</td>
<td>o</td>
</tr>
<tr>
<td>Record-Route</td>
<td>ar</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Reply-To</td>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>Require</td>
<td>ar</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>Retry-After</td>
<td>404,</td>
<td>ar</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>413,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>480,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>486</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>500,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>503</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>600,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>603</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Route</td>
<td>R</td>
<td>adr</td>
<td>o</td>
</tr>
<tr>
<td>Server</td>
<td>r</td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>Subject</td>
<td>R</td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>Timestamp</td>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>To</td>
<td>c(1)</td>
<td>r</td>
<td>m</td>
</tr>
<tr>
<td>Unsupported</td>
<td>420</td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>User-Agent</td>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>Via</td>
<td>R</td>
<td>amr</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>rc</td>
<td>dr</td>
<td>m</td>
</tr>
<tr>
<td>Warning</td>
<td>r</td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>401</td>
<td>ar</td>
<td>m</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>407</td>
<td>ar</td>
<td>o</td>
</tr>
</tbody>
</table>

(1): copied with possible addition of tag

Table 2: Summary of header fields, P--Z

A MESSAGE request MAY contain a body, using the standard MIME headers to identify the content.
11. Example Messages

An example message flow is shown in Figure 1. The message flow shows an initial IM sent from User 1 to User 2, both users in the same domain, "domain", through a single proxy.

```
<table>
<thead>
<tr>
<th>F1 MESSAGE</th>
<th>F2 MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>F4 200 OK</td>
<td>F3 200 OK</td>
</tr>
<tr>
<td>&lt;-----------------------------</td>
<td>&lt;-----------------------------</td>
</tr>
</tbody>
</table>

Figure 1: Example Message Flow

Message F1 looks like:

```
MESSAGE sip:user2@domain.com SIP/2.0
Via: SIP/2.0/UDP user1pc.domain.com
From: sip:user1@domain.com
To: sip:user2@domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 18

Watson, come here.
```

User1 forwards this message to the server for domain.com. The proxy receives this request, and recognizes that it is the server for domain.com. It looks up user2 in its database (built up through registrations), and finds a binding from sip:user2@domain.com to sip:user2@user2pc.domain.com. It forwards the request to user2. The resulting message, F2, looks like:
MESSAGE sip:user2@domain.com SIP/2.0
Via: SIP/2.0/UDP proxy.domain.com
Via: SIP/2.0/UDP user1pc.domain.com
From: sip:user1@domain.com
To: sip:user2@domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 18

Watson, come here.

The message is received by user2, displayed, and a response is generated, message F3, and sent to the proxy:

SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.domain.com
Via: SIP/2.0/UDP user1pc.domain.com
From: sip:user1@domain.com
To: sip:user2@domain.com;tag=ab8asdasd9
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Length: 0

Note that most of the header fields are simply reflected in the response. The proxy receives this response, strips off the top Via, and forwards to the address in the next Via, user1pc.domain.com, the result being message F4:

SIP/2.0 200 OK
Via: SIP/2.0/UDP user1pc.domain.com
From: sip:user1@domain.com
To: sip:user2@domain.com;tag=ab8asdasd9
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Length: 0

12. Security Considerations

In normal usage, most SIP requests are used to setup and modify communication sessions. The actual communication between participants happens in the media sessions, not in the SIP requests themselves. The MESSAGE method changes this assumption; MESSAGE requests normally carry the actual communication between participants as payload. This implies that MESSAGE requests have a greater need for security than most other SIP requests. In particular, UAs that support the MESSAGE request SHOULD support end-to-end authentication,
body integrity, and body confidentiality mechanisms.

12.1 Outbound authentication

When local proxies are used for transmission of outbound messages, proxy authentication is RECOMMENDED. This is useful to verify the identity of the originator, and prevent spoofing and spamming at the originating network.

12.2 SIPS URIs

The SIPS URI mechanism [1] allows a UA to assert that every hop must occur over a secure connection. This provides some level of integrity and privacy protection. However, this requires the users to trust that each proxy in the request path is well-behaved, that is, they do not violate the rules for routing SIPS URIs. Also, any unencrypted bodies are fully exposed to the proxies.

Additionally, the possibility of a forking proxy allows a MESSAGE request to be delivered to additional endpoints without the knowledge of the UAC. If only hop-by-hop protection is used, the users must trust all proxies in the chain to not fork requests to unauthorized destinations.

12.3 End-to-End Protection

UAs may provide end-to-end protection through the use of S/MIME. SIP allows the use of S/MIME to provide privacy and integrity protection of message bodies. S/MIME also allows privacy protection of SIP headers that are not read by proxies, and integrity protection of headers that are not modified by proxies.

Due to the greater security requirements for MESSAGE requests, UAs that support the MESSAGE method SHOULD support S/MIME.

12.4 Replay Prevention

To prevent the replay of old SIP requests, all signed MESSAGE requests and responses SHOULD contain a Date header covered by the message signature. Any message with a date older than several minutes in the past, or which is more than several minutes in the future, should be answered with a 400 (Incorrect Date or Time) message, unless such messages arrive repeatedly from the same source, in which case they MAY be discarded without sending a response. Obviously, this replay attack prevention mechanism does not work for devices without clocks.

Note that there are situations where an stale Date header is normal.
For example, the MESSAGE request may have been stored in a store and forward server while the recipient was offline. When the recipient returns, that server might then forward the message. Final receipt of the message would then occur some time after it was originally sent.

If a UAS receives a stale message that can be confirmed to have come from a known store and forward server (perhaps over a TLS connection), it makes sense for it to accept the message normally. Also, if one or more stale messages arrive shortly after an offline period, the UAS MAY accept the message, but SHOULD warn the user that there is a risk the message has been replayed.

12.5 Using message/cpim bodies

The message/cpim format [4] allows for the S/MIME protection of metadata in addition to the message payload itself. In many cases, this metadata is redundant with SIP headers. Still, message/cpim adds value in that the protection of metadata can extend across protocol boundaries. For example, a signed message/cpim body can provide sender authentication using the message/cpim From header, even if the message crosses a gateway to another CPIM compliant instant message service that does not understand SIP headers.

Therefore UAs SHOULD use the message/cpim format when protecting bodies using S/MIME. UAs may choose not to use message/cpim if they have knowledge that the message recipient, and all points between, are SIP devices.

13. IANA Considerations

This specification registers the MESSAGE method in the http://www.iana.org/assignments/sip-parameters/Method registry, according to the following information:

MESSAGE        [RFCXXXX]

14. Changes to This Document

14.1 Changes introduced in draft-ietf-sip-message-04

Added Scope of Applicability section to clarify the differences between pager-model and session-model IMs, and that this document only covers pager-model.

Strengthened the Congestion Control section.

Trimmed the author list.
Added Contributors section.

14.2 Changes introduced in draft-ietf-sip-message-03

Updated BNF to escape all characters in "MESSAGE". Fixed a few typos

14.3 Changes introduced in draft-ietf-sip-message-02

Updated references to the SIP specification.
Removed text that was redundant with SIP and CPIM documents.
Split references into normative and informational.
Added additional text on the issues of forking MESSAGE requests.
Added text on the meaning of 202 responses.
Updated tables 1 and 2 to reflect the current SIP specification.
Added IANA consideration section registering the MESSAGE method.
Removed terminology section because it was completely redundant with the SIP specification and RFC2779.
Added text to recommend that IM URIs be resolved as early as possible.
Removed discussion of using In-Reply-To for threading. This will be addressed in a separate "usage" draft, probably in the SIMPLE working group.
Removed analysis of RFC 2779 requirements--this may be moved to the usage draft.
Expanded the abstract section.
Removed "sales pitch" from the introduction.
Updated the Security Consideration section to include latest SIP security features.
Added text to Security Considerations concerning stale Date headers in offline messages.
Several editorial and organizational changes.
14.4 Changes introduced in draft-ietf-sip-message-01

The CPIM mapping section has been removed to a separate document. The references to the IMPP CPIM drafts have been updated to track newer versions.

14.5 Changed Introduced in draft-ietf-sip-message-00

The draft name changed (again) due to its move to the SIP working group.

The draft now clarifies that, while MESSAGE requests do not establish dialogs, user agents may group messages into conversation threads.

The draft clarifies the intend that all implementations must handle message/cpim body parts.

References to PGP encryption in SIP have been removed.

Open Issue concerning mapping between URI schemes at a CPIM compliant gateway device has been closed. This draft treats such mapping as a matter of local policy.

Added text for the congestion control section and removed related open issues.

14.6 Changes Introduced in draft-ietf-simple-im-01

This version removes the idea of implicit sessions created by MESSAGE requests. MESSAGE requests are now completely stateless in themselves.

The version also some open issues: Bodies are not allowed in responses; an Accept header on a 415 response includes body types nested inside message/cpim bodies, all IM UAs MUST be able to receive message/cpim.

This draft introduces a new section for CPIM mapping. The authors expect this section will need further work to complete.

14.7 Changes Introduced in draft-ietf-simple-im-00

The draft name changed to reflect its status as a SIMPLE working group item. This version introduces no other changes.

14.8 Changes Introduced in draft-rosenberg-impp-im-01

This submission serves to track transition of the work on a SIP
implementation of IM to the newly formed SIMPLE working group. It
endeavors to capture the progress made in IMPP since the original
submission (in particular, including the im: URI and the message/cpim
body) and detail a set of open issues for the SIMPLE working group to
address.

To support those goals, a great deal of the background and motivation
material in the original text has been shortened or removed.

15. Contributors

The following people made substantial contributions to this work:

Bernard Aboba     Microsoft
Steve Donovan     dynamicsoft
Jonathan Lennox   Columbia University
Dave Oran         Cisco
Robert Sparks     dynamicsoft
Dean Willis       dynamicsoft

16. Acknowledgments

The authors would like to thank the following people for their
support of the concept of SIP for IM, support for this work, and for
their useful comments and insights:

Jon Peterson     Neustar
Sean Olson       Microsoft
Adam Roach       dynamicsoft
Billy Biggs      University of Waterloo
Stuart Barkley   UUNet
Mauricio Arango  SUN
Richard Shockey  Neustar
Jorgen Bjorker   Hotsip
Henry Sinnreich  MCI Worldcom
Ronald Akers     Motorola
Torrey Searle    Indigo Software
Rohan Mahy       Cisco
Christian Groves Ericsson

Normative References

Protocol", draft-ietf-sip-rfc2543bis-09 (work in progress),
February 2002.


Informational References


Authors’ Addresses

Ben Campbell
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024
EMail: bcampbell@dynamicsoft.com

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
EMail: jdrosen@dynamicsoft.com
Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgement

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