

Network Working Group  
Request for Comments: 3428  
Category: Standards Track

B. Campbell, Ed.  
J. Rosenberg  
dynamicsoft  
H. Schulzrinne  
Columbia University  
C. Huitema  
D. Gurle  
Microsoft Corporation  
December 2002

## Session Initiation Protocol (SIP) Extension for Instant Messaging

### 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 (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.

This document proposes the MESSAGE method, an extension to the Session Initiation Protocol (SIP) that allows the transfer of Instant Messages. Since the MESSAGE request is an extension to SIP, it inherits all the request routing and security features of that protocol. 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.

## Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [6] and indicate requirement levels for compliant SIP implementations.

## Table of Contents

1.	Introduction . . . . .	2
2.	Scope of Applicability . . . . .	3
3.	Overview of Operation . . . . .	4
4.	UAC Processing . . . . .	5
5.	Use of Instant Message URIs . . . . .	6
6.	Proxy Processing . . . . .	6
7.	UAS Processing . . . . .	7
8.	Congestion Control . . . . .	8
9.	Method Definition . . . . .	9
10.	Example Messages . . . . .	11
11.	Security Considerations . . . . .	13
11.1	Outbound authentication . . . . .	13
11.2	SIPS URIs . . . . .	14
11.3	End-to-End Protection . . . . .	14
11.4	Replay Prevention . . . . .	14
11.5	Using message/cpim bodies . . . . .	15
12.	IANA Considerations . . . . .	15
13.	Contributors . . . . .	15
14.	Acknowledgments . . . . .	15
15.	Normative References . . . . .	16
16.	Informational References . . . . .	16
17.	Authors' Addresses . . . . .	17
18.	Full Copyright Statement . . . . .	18

## 1. Introduction

Instant Messaging (IM) 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 [11], 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.

RFC 2778 [3] and RFC 2779 [2] give a model and requirements for presence and instant messaging protocols. Implementations of the MESSAGE method SHALL support all the instant message requirements in RFC 2779 relevant to its scope of applicability.

## 2. Scope of Applicability

This document describes the use of the MESSAGE method for sending instant messages 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, if 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 authors 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 SHOULD 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 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 [7]. Since the message/cpim format is expected to be supported by other instant message protocols, endpoints using different IM protocols, but otherwise supporting message/cpim body types, should be able to

exchange messages without modification of the content by a gateway or other intermediary. This helps to enable end-to-end security between endpoints that use different IM protocols.

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) [8] 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 UACs which support the MESSAGE method MUST be prepared to send MESSAGE requests with a body of type text/plain. They may send bodies of type message/cpim [7].

MESSAGE requests do not initiate dialogs. User Agents MUST NOT insert Contact header fields 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 responses. Even though the UAC only sees one of those responses, the request has in fact been received by the second device as well.

The UAC MAY add an Expires header field to limit the validity of the message content. If the UAC adds an Expires header field with a non-zero value, it SHOULD also add a Date header field containing the time the message is sent.

## 5. Use of Instant Message URIs

An instant inbox may be most generally referenced by an Instant Message URI [8] in the form of "im:user@domain". IM URIs are abstract, and will eventually be translated to concrete, protocol-dependent URI.

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 header fields. 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 header fields MUST NOT contain IM URIs. These header fields 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]. 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 response to a MESSAGE request MUST NOT contain a body. A UAS MUST NOT insert a Contact header field into a 2xx 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 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 render bodies of type "text/plain", and may support reception and rendering of bodies of type "message/cpim" [7].

A MESSAGE request is said to be expired if its expiration time has passed. The expiration time is determined by examining the Expires header field, if present. MESSAGE requests without an Expires header field do not expire. If the MESSAGE request containing an Expires header field also contains a Date header field, the UAS SHOULD interpret the Expires header field value as delta time from the Date header field value. If the request does not contain a Date header field, the UAS SHOULD interpret the Expires header value as delta time from the time the UAS received the request.

If the MESSAGE expires before the UAS is able to present the message to the user, the UAS SHOULD handle the message based on local policy. This policy could mean: the message is deleted undisplayed,

the message is still displayed to the user, or some other policy may be invoked. If the message is displayed, the UAS SHOULD clearly indicate to the user that the message has expired.

If the UAS is acting as a message relay, and is unable to deliver the message before expiration, it chooses an action based on local policy. This action could involve deleting the message undelivered, delivering it as is, logging the expired message, or any other local policy.

## 8. 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 size of MESSAGE requests outside of a media session MUST NOT exceed 1300 bytes, unless the UAC has positive knowledge that the message will not traverse a congestion-unsafe link at any hop, or that the message size is at least 200 bytes less than the lowest MTU value found en route to the UAS. Larger payloads may be sent as part of a media session, or using some type of content-indirection.

At the time of this writing, there is no mechanism for a UAC to gain such knowledge outside of trivial network architectures, or networks that are wholly controlled by a single administrative domain. But if a mechanism for ensuring congestion-control at each hop is created in the future, MESSAGE clients can relax the size limit without requiring a change to this specification. The authors expect that such a mechanism or mechanism will be created in the near future.



There have been discussions on making the 1300 byte limit based on the path MTU to the next hop SIP device. The SIP specification does exactly that, choosing the limit 200 bytes less than the path MTU, or 1300 bytes if the device does not know the path MTU. Transport decisions are made on a per-hop basis. However, the point of this limit is to make sure that no upstream proxy chooses to send a MESSAGE request with large content over UDP. Since, except in trivial circumstances, a MESSAGE client is very unlikely to know the MTU for upstream devices beyond the next hop, an MTU based limit is not very useful.

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.

The prohibition against overlapping MESSAGE request provides some degree of congestion-safe behavior. A request and its associated response must each cross the full path between the UAC and the UAS. The time required for this will increase as networks become congested. This provides an adaptive mechanism to slow the introduction of new MESSAGE requests to the same destination.

It has been suggested that provisional responses should not be allowed for pager-model MESSAGE requests. However, it is not possible to require special treatment for MESSAGE, since many proxy servers will not be aware of the MESSAGE method. Therefore MESSAGE requests will receive the same provisional response treatment as any other non-INVITE method, as described in the SIP specification.

## 9. 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 header fields that can be used in MESSAGE requests and responses.

Header Field	where	proxy	MESSAGE
Accept	R		-
Accept	2xx		-
Accept	415		m*
Accept-Encoding	R		-
Accept-Encoding	2xx		-
Accept-Encoding	415		m*
Accept-Language	R		-
Accept-Language	2xx		-
Accept-Language	415		m*
Alert-Info	R		-
Alert-Info	180		-
Allow	R		o
Allow	2xx		o
Allow	r		o
Allow	405		m
Authentication-Info	2xx		o
Authorization	R		o
Call-ID	c	r	m
Call-Info		ar	o
Contact	R		-
Contact	1xx		-
Contact	2xx		-
Contact	3xx		o
Contact	485		o
Content-Disposition			o
Content-Encoding			o
Content-Language			o
Content-Length		ar	t
Content-Type			*
CSeq	c	r	m
Date		a	o
Error-Info	300-699	a	o
Expires			o
From	c	r	m
In-Reply-To	R		o
Max-Forwards	R	amr	m
Organization		ar	o

Table 1: Summary of header fields, A--O

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

(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 header fields to identify the content.

## 10. 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.

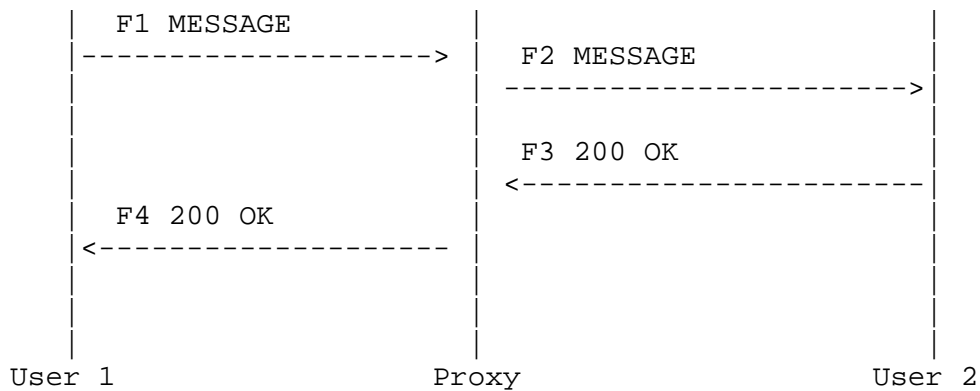


Figure 1: Example Message Flow

Message F1 looks like:

```

MESSAGE sip:user2@domain.com SIP/2.0
Via: SIP/2.0/TCP user1pc.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
From: sip:user1@domain.com;tag=49583
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/TCP proxy.domain.com;branch=z9hG4bK123dsghds
Via: SIP/2.0/TCP user1pc.domain.com;branch=z9hG4bK776sgdkse;
                                         received=1.2.3.4
Max-Forwards: 69
From: sip:user1@domain.com;tag=49394
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/TCP proxy.domain.com;branch=z9hG4bK123dsghds;
                                         received=192.0.2.1
Via: SIP/2.0/TCP user1pc.domain.com;;branch=z9hG4bK776sgdkse;
                                         received=1.2.3.4
From: sip:user1@domain.com;tag=49394
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/TCP user1pc.domain.com;branch=z9hG4bK776sgdkse;
                                         received=1.2.3.4
From: sip:user1@domain.com;;tag=49394
To: sip:user2@domain.com;tag=ab8asdasd9
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Length: 0
```

## 11. 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 MUST implement end-to-end authentication, body integrity, and body confidentiality mechanisms.

### 11.1 Outbound Authentication

When local proxies are used for transmission of outbound messages, proxy authentication, as specified in RFC 3261, is RECOMMENDED. This is useful to verify the identity of the originator, and prevent spoofing and spamming at the originating network.

## 11.2 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.

## 11.3 End-to-End Protection

When the goal is to remedy the concerns stated above, the MESSAGE bodies MUST be secured with S/MIME. If bodies specified in future to be carried by the MESSAGE method have different means of providing end-to-end security, their specification MUST describe the usage. SIP MESSAGE endpoints MUST support encryption (CMS EnvelopeData) and S/MIME signatures (CMS SignedData).

The S/MIME algorithms are set by RFC 3369 [4]. The AES [10] algorithm should be preferred, as it is expected that AES best suits the capabilities of many platforms. However, an IETF specification for this is still incomplete as of the time of this writing.

## 11.4 Replay Prevention

To prevent the replay of old SIP requests, all signed MESSAGE requests and responses MUST contain a Date header field 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 field 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.

### 11.5 Using message/cpim Bodies

The message/cpim format [7] allows for the S/MIME protection of metadata in addition to the message payload itself. In many cases, this metadata is redundant with SIP header fields. 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 field, even if the message crosses a gateway to another CPIM compliant instant message service that does not understand SIP header fields.

### 12. IANA Considerations

This specification registers the MESSAGE method in the [http://www.iana.org/assignments/sip-parameters/Method registry](http://www.iana.org/assignments/sip-parameters/Method%20registry), according to the following information:

MESSAGE	[RFC3428]
---------	-----------

### 13. 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

### 14. 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
Allison Mankin	ISI
Tan Ya-Ching	Siemens

## 15. Normative References

- [1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [2] Day, M., Aggarwal, S. and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.
- [3] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.
- [4] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.
- [5] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.
- [6] Bradner, S., "Keywords for Use in RFC's to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

## 16. Informational References

- [7] Atkins, D. and G. Klyne, "Common Presence and Instant Messaging Message Format", Work in Progress.
- [8] Crocker, D., Diacakis, A., Mazzoldi, F., Huitema, C., Klyne, G., Rose, M., Rosenberg, J., Sparks, R., Sugano, H. and J. Peterson, "Address Resolution for Instant Messaging and Presence", Work in Progress.
- [9] Rosenberg, J. and H. Schulzrinne, "SIP Caller Preferences and Callee Capabilities", Work in Progress.
- [10] Schaad, J. and R. Housley, "Use of the AES Encryption Algorithm and RSA-OAEP Key Transport in CMS", Work in Progress.



- [11] DellaFera, C., Eichin, M., French, R., Jedlinski, D., Kohl, J. and W. Sommerfeld, "The Zephyr notification service", in USENIX Winter Conference (Dallas, Texas), Feb. 1988.

## 17. Authors' Addresses

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

EMail: [bcampbell@dynamicsoft.com](mailto:bcampbell@dynamicsoft.com)

Jonathan Rosenberg  
dynamicsoft  
72 Eagle Rock Avenue  
First Floor  
East Hanover, NJ 07936

EMail: [jdrosen@dynamicsoft.com](mailto:jdrosen@dynamicsoft.com)

Henning Schulzrinne  
Columbia University  
M/S 0401  
1214 Amsterdam Ave.  
New York, NY 10027-7003

EMail: [schulzrinne@cs.columbia.edu](mailto:schulzrinne@cs.columbia.edu)

Christian Huitema  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052-6399

EMail: [huitema@microsoft.com](mailto:huitema@microsoft.com)

David Gurle  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052-6399

EMail: [dgurle@microsoft.com](mailto:dgurle@microsoft.com)

## 18. 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.

