AOL Time Warner

DOCKET FILE COPY ORIGINAL

Steven N. Teplitz
Vice President
Communications Policy
and Regulatory Affairs

RECEIVED

JUL 23 2001

FEDERAL COMMUNICATIONS COMMISSION
OFFICE OF THE SECRETARY

ORIGINAL

CS 00-30

July 23, 2001

Ms. Magalie Roman Salas
Secretary
Federal Communications Commission
445 12th Street, SW
Washington, DC 20554

Re: Progress Report on Instant Messaging Interoperability

Dear Ms. Salas:

Pursuant to the FCC's *Memorandum Opinion and Order* approving the merger between America Online, Inc. ("AOL") and Time Warner Inc. ("Time Warner"),[1] AOL Time Warner Inc. ("AOL Time Warner") hereby submits this progress report to update the Commission on AOL's ongoing efforts to develop a server-to-server IM interoperability solution that will allow a user of one of its IM services to exchange messages with users of unaffiliated IM services in a way that adequately protects IM network performance, privacy, and security.

---

[1] Memorandum Opinion and Order, *In the Matter of Applications for Consent to the Transfer of Control of Licenses and Section 214 Authorizations by Time Warner Inc. and America Online, Inc., Transferors, to AOL Time Warner Inc., Transferee*, CS Docket No. 00-30, FCC 01-12, ¶ 327 (rel. Jan. 22, 2001).

No. of Copies rec'd 0+4
List ABCDE

AOL Time Warner Inc. • 1101 Connecticut Avenue, NW, Suite 400 • Washington, DC 20036
tel 202 530 7883 • fax 202 530 7860 • steventep@aol.com

AOL publicly stated last July that it anticipated that it would require approximately one year to develop a server-to-server protocol, to be followed by a period of time to test and refine its interoperability solution. Consistent with this commitment, AOL has largely completed its development of the necessary technology, has recently begun internal testing of that technology, and remains on schedule to begin testing server-to-server interoperability with a leading technology company later this summer.

The Challenge Of IM Interoperability Is To Create A Safe And Secure Solution That Does Not Undermine The Essential Qualities That Have Made IM Popular

AOL attributes much of the success of its IM services to the qualities that distinguish these services from other forms of text-based communication: it is instant, it is reliable, and it is secure and private.[2]

- **Instant.** Messages and other communications are delivered quickly (*i.e.*, in near real-time), and users are notified immediately when their buddies sign on or off the service;[3]
- **Reliable.** IM systems perform at a high quality of service level and are designed to recognize and promptly address network failures (*e.g.*, PC

[2] Indeed, one of the biggest reasons for AOL's success in IM has been its vigilant approach, both in the design and day-to-day operations of its IM services, to protecting the user experience from disruptions, service outages, and/or security lapses that might jeopardize user confidence in its IM offerings.

[3] AOL's IM services today are specifically designed to ensure the prompt transmission of such data. For example, the AIM protocol is a binary protocol that provides more efficient data transmission than text-based protocols. In addition, AOL routes all server-to-server traffic within its IM networks on a private, high-speed LAN, thereby bypassing the threat to immediacy posed by data traffic congestion on the public Internet.

"crashes" and Internet traffic congestion);[4] and

- **Secure and Private**. IM services allow users to assume and control an identity (*i.e.*, a user name and password), and users are able to opt-out of messages they find intrusive.[5]

As explained below, interoperability, by definition, introduces a number of complicating issues that must be addressed in order to maintain these characteristics; otherwise, interoperability risks undermining the very reasons that IM has become as popular as it is today. As a result, it is not altogether surprising that to date others in the industry have yet to implement an interoperability solution, or that the IETF—while having made significant progress—has still not completed its work on server-to-server interoperability standards.

First, interoperability increases the potential for unacceptable delays in the transmission of messages and/or presence information, particularly across services.

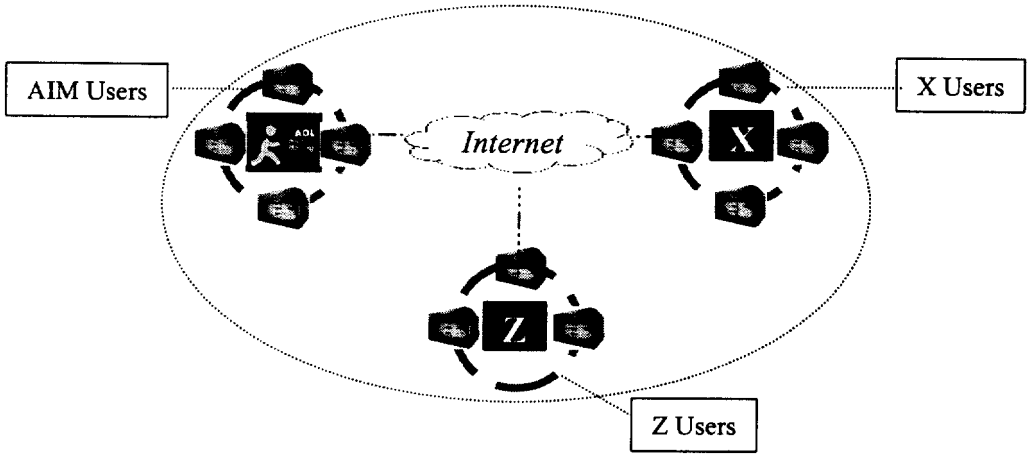
---

[4] The AIM network incorporates a number of safeguards designed to minimize threats to its reliability. For example, the AIM network includes hundreds of servers, including back-up servers that are constantly in "alert mode." Moreover, all of AOL's clients and servers communicate frequently to make sure that the connections between them are being maintained, and when AOL's IM clients detect a connection failure, they immediately notify users that they are no longer online.

[5] AOL's IM offerings have been specifically designed to provider users with a number of security and privacy features, including: (1) AIM's "knock-knock" feature, which, upon activation, requires user consent before displaying a message from a user not on their buddy list; (2) rate limits and user warnings, which impose limits on behavior within the AIM community; and (3) the IM feature of the AOL online service's "Notify AOL" function, which makes it possible to report offensive subscriber behavior directly to AOL.

By linking IM servers together, interoperability creates a single "virtual host" requiring continuous coordination and exchange of data between services:

"Virtual Host" System

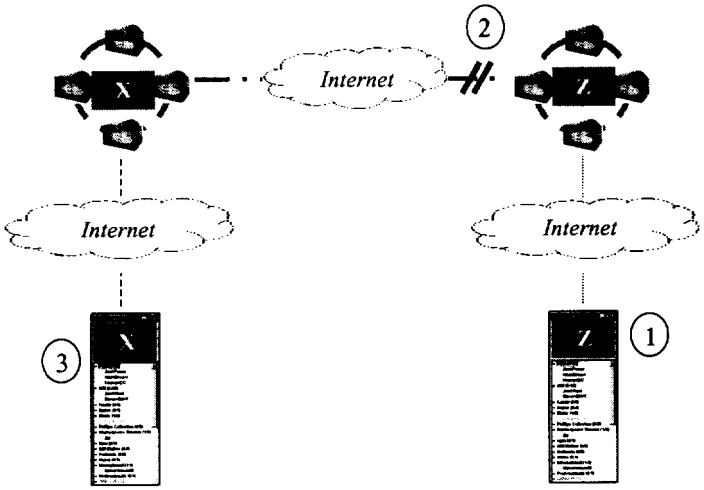


The problem with the "virtual host" approach, however, is that, to the extent that it relies upon the public Internet for the purpose of server-to-server communications, it potentially could lead to unacceptable delays in the transmission of message and presence data due to the data traffic congestion problems and bandwidth limitations that exist on the public Internet today.

Second, because interoperable IM providers will rely upon each other for accurate information, IM services will be affected by the service performance of all those systems with which they are interoperable—the reliable and unreliable alike. As is the case with email, IM systems participating in an interoperable network will operate to varying standards. Some potentially will suffer from poor performance and service outages. This is not a serious problem in email, because user expectations are more generous and the systems are designed to resend data whose receipt on the other end is not confirmed. In an interoperable IM network, however, failures will be difficult to identify and will

cascade inaccurate information throughout the IM systems participating in that network. As a result, the best performing systems could appear to be malfunctioning, potentially as often as those that are actually causing the problems. To illustrate:

Schematic of Interoperable Systems



1. User logs onto system Z, and X user sees him come online via presence notification.
2. Z system's Internet connection fails (internal Z system messages still work).
3. X user tries to IM Z user, but is unable to get message through (since Z system's connection has failed). Calls Z user who says he is still online. X user assumes X is at fault.

Thus, since IM services must rely upon each other for accurate presence and message information, outages will affect all systems—the reliable and unreliable alike.

Third, interoperability introduces potentially vulnerable points of access into IM providers' networks and forces IM providers to depend upon one another in their efforts to protect the privacy and security of their users. The points of vulnerability introduced by interoperability potentially enable bad actors, for example, to spam users with inappropriate images and/or text (*e.g.*, pornography), transmit viruses, impersonate IM users, or intercept messages. That is because interconnection points between two different networks, particularly if they are located on the public Internet, provide hackers with the opportunity to gain unauthorized access to those networks. In addition,

interoperability also requires an IM provider to rely upon others to help enforce its policies regarding harassment and other inappropriate conduct.

A viable interoperability approach must adequately address these concerns if it is to enhance the user experience rather than undermine IM's basic appeal. Moreover, if all of these concerns are not fully addressed from day one, there is no way to resolve them at a later date: once a flawed protocol has been implemented, it is virtually impossible—witness email—to undo the damage.

In light of these technical challenges, it is not surprising that none of the efforts others have initiated to allow users of different IM services to exchange messages has been successful to date.[6] Indeed, the IETF, the leading Internet standards-setting body, established the Instant Messaging and Presence Protocol ("IMPP") working group for the purpose of developing a single server-to-server IM interoperability standard. Last summer, however, the IMPP working group abandoned that goal due to its inability to reach consensus support for any single, comprehensive protocol, and has instead limited its efforts to developing common messaging formats which other working groups, subsequently formed by the IETF, are implementing as they develop several different

---

[6] One of these initiatives was launched—during the height of the IM debate before the FCC last summer—by IMUnified. Originally, IMUnified announced that it would "provide a basic framework for detailing the mechanics of IM exchange among our members systems by the end of August [2000], with final implementation across member communities expected by the end of [2000]." *See* IMUnified FAQ <http://www.imunified.org/faq.html>. Both of those deadlines have since passed unmet.

server-to-server interoperability protocols.[7] At this point, there is no announced timetable indicating when the efforts to develop those protocols will be completed.

AOL Has Made Significant Progress Toward Developing A Server-To-Server Interoperability Solution, Has Recently Begun Internal Testing, And Is On Schedule To Conduct A System-To-System Trial With A Leading Technology Company

Last July, AOL publicly stated that it would require approximately twelve months from that date to develop a server-to-server IM interoperability protocol, plus an additional testing period to ensure that that protocol will not undermine AOL's continued ability to protect its IM users' experience from the types of risks described above. Consistent with this commitment (and despite the challenges described above), AOL has assembled the technology necessary to exchange messages and presence information between IM networks, has recently begun internal testing of that technology, and remains on schedule to begin testing server-to-server interoperability with a leading technology company by late Summer 2001.

On July 15, 2000, AOL submitted a white paper to the IETF outlining its proposed framework for server-to-server interoperability. Subsequently, additional working groups were formed within the IETF to implement a number of divergent

---

[7] The IMPP working group is working on the following Internet-drafts defining common messaging formats: "Common Presence and Instant Messaging: Message Format," <http://www.ietf.org/internet-drafts/draft-ietf-impp-cpim-msgfmt-03.txt>; and "Date and Time on the Internet: Time Stamps," <http://www.ietf.org/internet-drafts/draft-ietf-impp-datetime-04.txt>. Copies of these documents are attached.

approaches to server-to-server interoperability.[8] In addition, other server-to-server interoperability efforts have been initiated in the IM marketplace, including open-source projects. AOL has evaluated each of these approaches with respect to its ability to satisfy AOL's requirements—in essence, whether it is capable of ensuring that IM's instant, reliable, and secure and private qualities survive the transition from an environment where a single provider controls the IM network from end to end to an environment in which IM providers depend upon the performance of all other providers' networks with which they are interoperable.

In the end, AOL determined that the optimal approach would be to develop a server-to-server interoperability framework using one of the standards being developed by the IETF. Of those, AOL selected the protocol being developed by the IETF's SIP for Instant Messaging and Presence Leverage ("SIMPLE") working group, which is working on an IM-specific implementation of the IETF's telephony-oriented Session Initiation Protocol ("SIP"). Among its considerations, AOL found that SIMPLE (and/or the SIP protocol from which it is derived) is already supported by a number of hardware and software companies and has a significant following among developers. The IETF Internet-draft describing in technical detail the SIMPLE messaging protocol, "SIP

---

[8] As noted above, the IETF originally chartered a single working group, the IMPP working group, to develop a single IM server-to-server Internet standard. Because that working group was unable to achieve consensus support for any single protocol, three additional working groups—APEX, PRIM, and SIMPLE—were established to pursue divergent approaches to server-to-server interoperability. To date, none of these working groups has finished specifying its protocol.

Extensions for Instant Messaging," is attached hereto and is also available at http://search.ietf.org/internet-drafts/draft-ietf-simple-im-00.txt; "SIP Extensions for Presence," the IETF Internet-draft describing in technical detail the SIMPLE presence protocol, is attached hereto and is also available at http://search.ietf.org/internet-drafts/draft-ietf-simple-presence-00.txt.

Because the SIMPLE working group has not finalized these protocols, however, AOL has had to resolve certain unsettled issues in the few functional areas where the working group has yet to make its final decisions. In particular, the comprehensive approach to interoperability AOL is working to complete will specify:

- That IM systems may establish dedicated, high-speed connections between their networks, thereby minimizing any bandwidth-related threats to the "instant" nature of IM;
- A quality of service level to which participating systems shall perform; and
- A standardized approach to privacy and security, including measures to protect users from spam and harassment.

Having thus assembled the components necessary to achieve basic interoperability—*i.e.*, the exchange of presence and message data—with other providers' IM systems, AOL is working to address additional implementation issues that must be resolved before it can introduce its interoperability solution into a real-world environment. At the same time, AOL is currently testing its basic interoperability components internally and is preparing to begin testing its comprehensive interoperability solution with an external partner.

To this end, AOL had to first develop an interoperable version of each component of the AIM service. This involved:

- creating a new version of the AIM client software;
- incorporating the ability to accept and process presence and message information from non-AOL systems into the AIM servers; and
- developing a gateway to translate the internal AIM protocol into the SIMPLE protocol in order to enable communication with other servers.

AOL completed this work in early July, and AOL has since been conducting internal trials intended to confirm its ability to pass presence and message information successfully between two model IM networks.

Once internal testing is completed, AOL intends to conduct a trial of its comprehensive interoperability solution, and is close to finalizing an agreement with a leading technology company that will allow the two companies to conduct a live server-to-server interoperability trial. In addition, AOL is working with this potential partner to draft a contractual agreement that addresses such concerns as performance requirements, cost sharing, and privacy and security policies. Upon successful completion of these tasks, AOL then plans to finalize its gateway, install updated code on its production servers, and begin developing a finished client that supports interoperability.

* * *

We appreciate this opportunity to have updated the Commission on AOL's progress on IM interoperability.

Respectfully submitted,

Steven N. Teplitz
Vice President, Communications Policy
And Regulatory Affairs
AOL Time Warner Inc.

cc: Chairman Michael K. Powell
Commissioner Gloria Tristani
Commissioner Kathleen Q. Abernathy
Commissioner Michael J. Copps
Commissioner Kevin J. Martin
W. Kenneth Ferree, Chief, Cable Services Bureau

Attachments:

"SIP Extensions for Instant Messaging"
"SIP Extensions for Presence"
"Common Presence and Instant Messaging: Message Format"
"Date and Time on the Internet: Time Stamps"

Internet Engineering Task Force
Internet-Draft
Expires: October 11, 2001

J. Rosenberg
D. Willis
R. Sparks
B. Campbell
dynamicsoft
H. Schulzrinne
J. Lennox
Columbia University
C. Huitema
B. Aboba
D. Gurle
Microsoft Corporation
D. Oran
Cisco Systems
April 12, 2001

SIP Extensions for Instant Messaging
draft-ietf-simple-im-00

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 October 11, 2001.

Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

This document defines a SIP extension (a single new method) that

supports Instant Messaging (IM).

Table of Contents

1. Introduction 4
2. Changes Introduced in draft-ietf-simple-im-00 4
3. Changes Introduced in draft-rosenberg-impp-im-01 5
4. Terminology 5
5. Overview of Operation 5
6. The MESSAGE request 6
6.1 Method Definition 6
6.2 UAC processing of initial MESSAGE request 8
6.3 Finding the next hop 9
6.4 Proxy processing of MESSAGE requests 9
6.5 UAS processing of MESSAGE requests 10
6.6 UAS processing of initial MESSAGE response 10
6.7 Subsequent MESSAGE requests 11
6.8 Supporting multiple message destinations 11
6.9 Caller Preferences 12
6.10 Security 12
6.10.1 Privacy 12
6.10.2 Message Integrity and Authenticity 13
6.10.3 Outbound authentication 13
6.10.4 Replay Prevention 14
7. Congestion Control 14
8. Example Messages 14
9. Open Issues 17
9.1 Must a MESSAGE actually include a message? 17
9.2 Should support for message/cpim be mandatory in all UAs? . 18
9.3 message/cpim and the Accept header 18
9.4 Message Sessions 18
9.5 What would a body in a 200 OK to a MESSAGE mean? 18
9.6 The im: URL and RFC2543 proxies and registrars 19
9.7 Providing im: URL in Contact headers 19
9.8 Congestion control 19
9.9 Mapping to CPIM 19
10. Acknowledgements 19
References 20
Authors' Addresses 21
A. Requirements Evaluation 23
Full Copyright Statement 27

1. Introduction

This document defines an extension to SIP (RFC2543 [2]) to support Instant Messaging.

Instant messaging is defined as the exchange of content between a set of participants in real time. Generally, the content is short

textual 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 [1], 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. Furthermore, most of these protocols tightly couple presence and IM, due to the way in which the service is offered.

Despite the popularity of presence coupled IM services, IM is a separate application from presence. There are many ways to use IM outside of presence (for example, as part of a voice communications session). Another example are interactive games (possibly established with SIP - SIP can establish any type of session, not just voice or video); IM is already a common component of multiplayer online games. Keeping it apart from presence means it can be used in such ways. Furthermore, keeping them separate allows separate providers for IM and for presence service. Of course, it can always be offered by the same provider, with both protocols implemented into a single client application.

Along a similar vein, the mechanisms needed in an IM protocol are very similar to those needed to establish an interactive session - rapid delivery of small content to a user at their current location, which may, in general, be dynamically changing as the user moves. The similarity of needed function implies that existing solutions for initiation of sessions (namely, the Session Initiation Protocol (SIP) [2]) is an ideal base on which to build an IM protocol.

2. 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.

3. 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: URL 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.

4. Terminology

Most of the terminology used here is defined in RFC2778 [4]. However, we duplicate some of the terminology from SIP in order to clarify this document:

User Agent (UA): A UA is a piece of software which is capable of initiating requests, and of responding to requests.

User Agent Server (UAS): A UAS is the component of a UA which receives requests, and responds to them.

User Agent Client (UAC): A UAC is the component of a UA which sends requests, and receives responses.

Registrar: A registrar is a SIP server which can receive and process REGISTER requests. These requests are used to construct address bindings.

5. 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 im: URL of the party to whom the message is directed (see CPIM [15]), but can also be a normal SIP URL. The body of the request will contain the message to be delivered. This body can be of any MIME type, including "message/cpim" [16].

The request may traverse a set of SIP proxies using a variety of transport mechanism (UDP, TCP, even SCTP [5]) before reaching its destination. The destination for each hop is located using the address resolution rules detailed in the CPIM and SIP specifications (see Section 6 for more detail). 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.

Groups of messages in a common thread may be associated by keeping them in the same session as identified by the combination of the To, From and Call-ID headers. Other potential means of grouping messages are discussed below.

It is possible that a proxy may fork a MESSAGE request based on its available routing mechanism. This draft proposes a mechanism that takes advangage of this, delivering messages in a session to multiple endpoints until one sends a message back. After that, all

remaining messages in the session are delivered to the responding agent.

6. The MESSAGE request

This section defines the syntax and semantics of this extension.

6.1 Method Definition

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

```
Message  =  "MESSAGE"
```

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

Tables 1 and 2 extend Tables 4 and 5 of SIP by adding an additional column, defining the headers that can be used in MESSAGE requests and responses.

	where	enc.	e-e	MESSAGE
Accept	R		e	o
Accept	415		e	o
Accept-Encoding	R		e	o
Accept-Encoding	415		e	o
Accept-Language	R		e	o
Accept-Language	415		e	o
Allow	200		e	o
Allow	405		e	m
Authorization	R		e	o
Authorization	r		e	o
Call-ID	gc	n	e	m
Contact	R		e	m
Contact	2xx		e	o
Contact	3xx		e	o
Contact	485		e	o
Content-Encoding	e		e	o
Content-Length	e		e	m

Content-Type	e		e	*
CSeq	gc	n	e	m
Date	g		e	o
Encryption	g	n	e	o
Expires	g		e	o
From	gc	n	e	m
Hide	R	n	h	o
Max-Forwards	R	n	e	o
Organization	g	c	h	o

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

	where	enc.	e-e	MESSAGE
Priority	R	c	e	o
Proxy-Authenticate	407	n	h	o
Proxy-Authorization	R	n	h	o
Proxy-Require	R	n	h	o
Record-Route	R		h	o
Record-Route	2xx,401,484		h	o
Require	R		e	o
Retry-After	R	c	e	-
Retry-After	404,413,480,486	c	e	o
	500,503	c	e	o
	600,603	c	e	o
Response-Key	R	c	e	o
Route	R		h	o
Server	r	c	e	o
Subject	R	c	e	o
Timestamp	g		e	o
To	gc(1)	n	e	m
Unsupported	420		e	o
User-Agent	g	c	e	o
Via	gc(2)	n	e	m
Warning	r		e	o

WWW-Authenticate	R	c	e	o
WWW-Authenticate	401	c	e	o

(1): copied with possible addition of tag
(2): UAS removes first Via header field

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

A MESSAGE request MAY (Open Issue Section 9.1) contain a body, using the standard MIME headers to identify the content.

Unless stated otherwise in this document, the protocol for emitting and responding to a MESSAGE request is identical to that for a BYE request as defined in [2]. The behavior of SIP entities not implementing the MESSAGE (or any other unknown) method is explicitly defined in [2].

6.2 UAC processing of initial MESSAGE request

A MESSAGE request MUST contain a To, From, Call-ID, CSeq, Via, Content-Length, and Contact header, formatted as specified in [2].

All UAs MUST be prepared to send and receive MESSAGE requests with a body of type text/plain. All UAs wishing to provide the end to end security mechanisms defined in CPIM MUST be prepared to send and receive MESSAGE requests with a body type of message/cpim. All UAs

implementing MESSAGE SHOULD provide the end to end security mechanisms defined in CPIM (Open Issue Section 9.2).

MESSAGE requests MAY contain an Accept header listing the allowable MIME types which may be sent in the response, or in subsequent requests in the reverse direction. The absence of the Accept header implies that the only allowed MIME type is text/plain. This simplifies operation in small devices, such as wireless appliances, which will generally only have support for text, but still allows any other MIME type to be used if both sides support it. (Open Issue Section 9.3)

A UAC MAY send a MESSAGE request within an existing SIP call, established with an INVITE. In this case, the MESSAGE request is processed identically to the INFO method [9]. The only difference is that a MESSAGE request is assumed to be for the purpose of instant messaging as part of the call, whereas INFO is less specific.

A UAC MAY associate sequential MESSAGEs in a common thread by constructing them with common To, From, and Call-ID headers and increasing CSeq values. (Open Issue Section 9.4)

6.3 Finding the next hop

The mechanism used to determine the next hop destination for a SIP MESSAGE request is detailed in [15] and [2]. Briefly, for the URL im:user@host,

1. The UA makes a DNS SRV [12] query for _im._sip.host. If any RRs are returned, they determine the next hop. Otherwise:
2. The UA makes a DNS SRV query for _sip.host. If any RRs are returned, they determine the next hop. Otherwise:
3. The UA makes a DNS A query for host. If any records are returned, they determine the address of the next hop. The desination port is determined from the input URL (if the input was an im: URL, the request is sent to the default SIP port of 5060).

For sip: URLs, the UA starts at step 2.

6.4 Proxy processing of MESSAGE requests

Proxies route requests with method MESSAGE the same as they would any other SIP request (proxy routing in SIP does not depend on the method). Note that the MESSAGE request MAY fork; this allows for delivery of the message to several possible terminals where the user might be.

If a MESSAGE request hits a proxy that uses registrations to route requests, but no registration exists for the target user in the request-URI, the request is rejected with a 404 (Not Found).

Proxies MAY have access rules which prohibit the transmission of instant messages based on certain criteria. Typically, this criteria will be based on the identity of the sender of the instant messages. Establishment of this criteria in the proxy is outside the scope of this extension. We anticipate that such access controls will often be controlled through web pages accessible by users, mitigating the need for standardization of a protocol for defining access rules.

6.5 UAS processing of MESSAGE requests

As specified in RFC 2543, if a UAS receives a request with a body of type it does not understand, it MUST respond with a 415 (Unsupported Media Type) containing an Accept header listing those types which are acceptable. (This brings up Open Issue Section 9.3 again)

Servers MAY reject requests (using a 413 response code) that are too long, where too long is a matter of local configuration. All servers MUST accept requests which are up to 1184 bytes in length.

> 1184 = minimum IPv6 guaranteed length (1280 bytes) minus UDP (8 bytes) minus IPSEC (48 bytes) minus layer one encapsulation (40 bytes).

A UAS receiving a MESSAGE request SHOULD respond with a final response immediately. A 200 OK is sent if the request is acceptable. Note, however, that the UAS is not obliged to display the message to the user either before or after responding with a 200 OK. A 200 class response to a MESSAGE request MAY contain a body, but this will often not be the case, since these responses are generated automatically. (Open Issue Section 9.5)

Like any other SIP request, an IM MAY be redirected, or otherwise responded to with any SIP response code. Note that a 200 OK response to a MESSAGE request does not mean the user has read the message.

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) response indicating that the message was accepted, but end to end delivery has not been guaranteed.

6.6 UAS processing of initial MESSAGE response

A 200 OK response to an initial IM may contain Record-Route headers. If present, these MUST be used to construct a Route header for use in subsequent requests for the same call-leg (defined as the combination of remote address, local address, and Call-ID), using the process described in Section 6.29 of SIP [2] as if the request were INVITE. Note that per Section 6.8 the 200 OK response may not contain a Contact header.

A 400 or 500 class response indicates that the message was not delivered successfully. A 600 response means it was delivered successfully, but refused.

6.7 Subsequent MESSAGE requests

Any subsequent MESSAGEs in a session (see Section 9.4 follow the path established by the Route headers computed by the UA. The CSeq header MUST be larger than a CSeq header used in a previous request for the same call leg. Is is strongly RECOMMENDED that the CSeq number be computed as described in Section 6.17 of SIP, using a clock. This allows for the CSeq to increment without requiring the UA to store the previous CSeq values.

6.8 Supporting multiple message destinations

A UAS MAY include a Contact in a 200 class response. Including a Contact header enables end to end messaging, which is good for efficiency. However, it rules out the possibility of effectively supporting more than one terminal which can handle IM simultaneously.

> This odd but seemingly innocuous requirement enables a very important feature. If a user is connected at several hosts, an initial IM will fork, and arrive at each. Each UAS responds with a 200 OK immediately, one of which is arbitrarily forwarded upstream towards the UAC. If another IM is sent for the same call-leg, we still wish for this IM to fork, since we still don't know where the user is currently residing. This information is known when the user sends an IM in the reverse direction. This IM will contain a Contact, and when it arrives at the originator of the initial MESSAGE, will update the Route so that now IMs are delivered only to that one host where the user is residing.

A UAS constructs a set of Route headers from the Record-Route and

Contact headers in the MESSAGE request, as per the procedure defined in [10].

MESSAGE requests for an established IM session MUST contain a Tag in the From field. Responses to an IM SHOULD contain a tag in the To field. This represents a slightly different operation than for INVITE. When a user sends an INVITE, they will receive a 200 OK with a tag. Requests in the reverse direction then contain that tag, and that tag only, in the From field. Here, the response to IM will contain a tag in the To field, and a MESSAGE will contain a tag in the From field. However, the UA may receive MESSAGE requests with tags in the From field that do not match the tag in the 200 OK received to the initial IM. This is because only a single 200 OK is returned to a MESSAGE request, as opposed to multiple 200 OK for

INVITE. Thus, the UA MUST be prepared to receive MESSAGEs with many different tags, each from a different PUA.

A UAS MUST be prepared to update the Route is has stored for an IM session with a Contact received in a request, if that Contact is different from one previously received, or if there was no Contact previously.

6.9 Caller Preferences

User agents SHOULD add the "methods" tag defined in the caller preference specification [8] to Contact headers with SIP URLs 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 [8].

6.10 Security

End-to-end security concerns for instant messaging were a primary driving force behind the creation of message/cpim [16]. Applications needing end-to-end security should study that work carefully.

SIP provides numerous security mechanisms which can be utilized in addition to those made available through the use of message/cpim.

6.10.1 Privacy

In order to enhance privacy of instant messages, it is RECOMMENDED that between network servers (proxies to proxies, proxies to redirect servers), transport mode ESP [6] or TLS is used to encrypt all traffic. Coupled with persistent connections, this makes it impossible for eavesdroppers on non-UA connections to determine when a particular user has even sent an IM, let alone what the content is. Of course, the content of unencrypted IMs are exposed to proxies.

Between a UAC and its local proxy, TLS [11] is RECOMMENDED. Similarly, TLS SHOULD be used between a proxy and the UAS receiving the IM. The proxy can determine whether TLS is supported by the receiving client based on the transport parameter in the Contact header of its registration. If that registration contains the token "tls" as transport, it implies that the UAS supports TLS. (Open issue Section 9.7)

Furthermore, we allow for the Contact header in the MESSAGE request to contain TLS as a transport. The Contact header is used to route subsequent messages between a pair of entities. It defines the address and transport used to communicate with the user agent for subsequent requests in the reverse direction. If no proxies insert Record-Route headers, the recipient of the original IM, when it wishes to send an IM back, will use the Contact header, and establish a direct TLS connection for the remainder of the IM communications. If a proxy does Record-Route, the situation is different. When the recipient of the original IM (call this participant B) sends an IM back to the originator of the original IM (call this participant A), this will be sent to the proxy closest to B which inserted Record- Route. This proxy, in turn, sends the request to the proxy before it which Record-Routed. The first proxy after A which inserted Record- Route will then use TLS to contact A. Since we suspect that most proxies will not insert Record-Route into instant messages, efficient, secure, direct IM will occur frequently.

If encrypted message/cpim bodies are not available, sensitive data may be protected from being observed by intermediate proxies by using SIP encryption for the transmission of MESSAGE requests. SIP supports PGP based encryption, which does not require the establishment of a session key for encryption of messages within a session (basically, a new session key is established for each message as part of the PGP encryption).

6.10.2 Message Integrity and Authenticity

In addition to the integrity and authenticity protections offered through message/cpim, SIP provides PGP based authentication and message integrity checks (both challenge-response and normal signatures), as well as http basic and digest authentication.

6.10.3 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.

6.10.4 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.

Furthermore, all signed SIP MESSAGE requests MUST contain a Call-ID and CSeq header covered by the message signature. A user agent MAY store a list of Call-ID values, and for each, the higest CSeq seen within that Call-ID. Any message that arrives for a Call-ID that exists, whose CSeq is lower than the highest seen so far, is discarded.

Finally, challenge-response authentication MAY be used to prevent replay protection.

7. Congestion Control

(Open Issue Section 9.8) Discussion needs to take place to populate this section.

8. 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. A second IM, sent in response, flows directly from User 2 to User 1.

```
|  F1 MESSAGE                |                          |
|-------------------->       |  F2 MESSAGE              |
|                            | ----------------------->|
|                            |                          |
|                            |  F3 200 OK               |
|                            | <-----------------------|
|  F4 200 OK                 |                          |
|<--------------------       |                          |
|                            |                          |
|                            |                          |
|                            |                          |
```

```
          |                      |   F5 MESSAGE            |
          | <--------------------|------------------------ |
```



```
          |                      |                         |
          |     F6 200 OK        |                         |
          | ---------------------|------------------------>|
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |
          |                      |                         |

      User 1                  Proxy                    User 2
```

Figure 1: Example Message Flow

Message F1 looks like:

```
MESSAGE im:user2@domain.com SIP/2.0
Via: SIP/2.0/UDP user1pc.domain.com
From: im:user1@domain.com
To: im:user2@domain.com
Contact: sip:user1@user1pc.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 (discovered through the combination of SRV and A record processing described in Section 6.3 , using UDP. 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 im:user2@domain.com to sip:user2@user2pc.domain.com. It forwards the request to user2, and does not insert a Record-Route header. 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: im:user1@domain.com
To: im:user2@domain.com
Contact: sip:user1@user1pc.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: im:user1@domain.com
To: im:user2@domain.com;tag=ab8asdasd9
Contact: sip:user2@user1pc.domain.com
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: im:user1@domain.com
To: im:user2@domain.com;tag=ab8asdasd9
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Length: 0
```

Now, user2 wishes to send an IM to user1, message F5. As there are no Record-Routes in the original IM, it can simply send the IM directly to the address in the Contact header. Note how the To and From fields are now reversed from the response it sent in message F4: