Internet Engineering Task Force
Internet-Draft
Expires: January 16, 2002

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
July 18, 2001

SIP Extensions for Instant Messaging
draft-ietf-simple-im-01

Status of this Memo

Copyright Notice

Abstract

This document defines a SIP extension (a single new method) that supports Instant Messaging (IM).

Table of Contents

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-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.

3. 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.

4. 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.

5. 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.

6. 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 7 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.

MESSAGE requests do not create any implied session. They do not in themselves establish a call leg, or any concept of call state. SIP proxies may not record-route MESSAGE requests.

7. The MESSAGE request

This section defines the syntax and semantics of this extension.

7.1 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	-
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	-
Contact	2xx		e	-
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	-
Record-Route	2xx,401,484		h	-
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 13.1) 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].

7.2 UAC processing of MESSAGE request

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

All UAs MUST be prepared to send and receive MESSAGE requests with a body of type text/plain. They MUST be prepared to receive message/cpim body types, and MAY choose to send message/cpim body types.

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.

MESSAGE requests do not imply any sort of association or session on their own. User Agents MUST not insert contact headers into MESSAGE requests.

A UA SHOULD NOT attempt to associate REQUEST messages with each other in any way, unless those messages are part of a call leg established through other means.

7.3 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 destination 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.

7.4 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.

Proxies MUST NOT create call state based on MESSAGE requests alone. Proxies MUST NOT insert record-route headers. If a route header is present in the request, a proxy MUST honor it.

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.

7.5 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 list SHOULD also include types acceptable nested within a message/cpim body.

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 MUST NOT contain a body.

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.

7.6 UAS processing of MESSAGE response

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

A UAS MUST NOT insert a contact header into a 200 class response.

8. 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].

9. Mapping to CPIM

9.1 Mapping SIP Requests to CPIM

The CPIM draft[15] describes an abstract set of Instant Messaging operations that all instant messaging services must map to to insure interoperability. This section describes the mapping between SIP instant messaging and CPIM.

The SIP MESSAGE request maps to the CPIM message operation, which requires the parameters of source, destination, transID, and the message to be sent.

The From header maps to the source parameter, in the case of unauthenticated MESSAGE requests. However, a CPIM compliant gateway SHOULD authenticate the message request. If the request is in fact authenticated, then the source parameter MUST be the authenticated credentials of the sender.

The requestURI maps to the destination parameter.

The transID mapping needs further discussion (Open Issue Section 13.5.) It must contain enough information to identify the transaction state at a CPIM gateway, so that the correct SIP response can be generated.

The message to be sent maps to the body of the MESSAGE request. If the body has a MIME type of message/cpim, it SHOULD be sent as is. Any other mime-types MUST be embedded into a message/cpim body part.

If a gateway UAS cannot determine the results of the message operation in a short time, it MUST return a 202 accepted response.

9.2 Mapping CPIM Responses to SIP

CPIM specifies the response types of success, failure, and indeterminate. A success response maps to a 200 OK message. An indeterminate response maps to a 202 Accepted response. The mapping for failure needs further discussion. (Open Issue Section 13.5)

9.3 Mapping CPIM operations to SIP

When a gateway maps a CPIM message operation to SIP, it MUST generate a MESSAGE request. The request URI and the To header MUST both contain the URL from the CPIM destination parameter. The From header SHOULD contain the URL from the source parameter. The message MUST copied into the body unchanged. Otherwise, the MESSAGE request is generated using normal SIP. The gateway UAC MUST keep enough transaction state to be able to determine the transID from the SIP response.

9.4 Mapping SIP responses to CPIM

Then the gateway UAC receives a response to a MESSAGE request, it determines the CPIM response according to the following rules: A 202 response maps to "undetermined." Any other 200 class response maps to "success." Any 400, 500, and 600 class response maps to "failure". 100 class responses MUST be consumed by the gateway UAC. 300 class responses SHOULD be acted upon by the gateway UAC according to normal SIP rules.

9.5 URL Scheme Mapping

Mapping of URLs between URL schemes needs further discussion.(Open Issue Section 13.5) This is likely to be controlled by local policy at least to some degree.

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.

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 13.3)

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).

10.2 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.

10.3 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 highest 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.

11. Congestion Control

(Open Issue Section 13.4) Discussion needs to take place to populate this section.

12. 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.

```
       |  F1 MESSAGE           |                          |
       |---------------------> |  F2 MESSAGE              |
       |                       | ------------------------>|
       |                       |                          |
       |                       |  F3 200 OK               |
       |                       | <------------------------|
       |  F4 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
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 7.3 , 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. 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
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
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
```

13. Open Issues

13.1 Must a MESSAGE actually include a message?

Section 7 specifies that a MESSAGE MAY contain a MIME body. Should this be MUST? Does it make sense to have a MESSAGE with no body?

13.2 The im: URL and RFC2543 proxies and registrars

What are the implications of an im: URL showing up in the request URI in a MESSAGE request received by an RFC2543 proxy, or the To: header of a REGISTER request received by an RFC2543 registrar?

13.3 Providing im: URL in Contact headers

What are the ramifications of a UA providing an im: URL in a Contact: header for a REGISTER method, or a MESSAGE method? For the forseeable future, most SIP endpoints aren't going to have SRV records of the form _im._sip.host or even _sip.host pointing to them. Falling back to A records in that case seems to preclude the use of non-UDP transports.

13.4 Congestion control

Per the amendments made to the SIMPLE charter by the IESG prior to approval, congestion control needs attention. In particular the requirements of BCP 41 must be met by this extension. Specifying the use of transport protocols with congestion control built in, particularly with the recommendation of reuse of connections, is an option. The question is when can we use those that don't (UDP) and what needs to be done in addition to what SIP already does in that case. Among other things, this interacts with Section 13.3

13.5 Mapping to CPIM

This version offers a first cut at describing CPIM mapping. However, the entire subject needs further discussion. How do we map SIP transactions to CPIM "transID?" What is the correct SIP response code for a CPIM failure response? How do we handle mapping between URL schema at a gateway? Do we need to describe gateway timing issues?

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   Level(3) Communications
Sean Olson   Ericsson
Adam Roach   Ericsson
Billy Biggs   University of Waterloo
Stuart Barkley   UUNet
Mauricio Arango SUN
Richard Shockey Shockey Consulting LLC
Jorgen Bjorker   Hotsip
Henry Sinnreich MCI Worldcom
Ronald Akers   Motorola

References

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

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

[3] Day, M., Aggarwal, S. and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[4] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[5] Rosenberg, J. and H. Schulzrinne, "SCTP as a transport for SIP", draft-rosenberg-sip-sctp-00 (work in progress), June 2000.

[6] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[7] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[8] Rosenberg, J. and H. Schulzrinne, "SIP caller preferences and callee capabilities", draft-ietf-sip-callerprefs-03 (work in progress), November 2000.

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

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

[11] Dierks, T., Allen, C., Treese, W., Karlton, P. L., Freier, A. O. and P. C. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[12] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[13] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[14] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[15] Crocker, D., Diacakis, A., Mazzoldi, F., Huitema, C., Klyne, G., Rose, M., Rosenberg, J., Sparks, R. and H. Sugano, "A Common Profile for Instant Messaging (CPIM)", draft-ietf-impp-cpim-01 (work in progress), February 2001.

[16] Atkins, D. and G. Klyne, "Common Presence and Instant Messaging Message Format", draft-ietf-impp-cpim-msgfmt-00 (work in progress), February 2001.

Authors' Addresses

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

email: jdrosen@dynamicsoft.com

Dean Willis
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

email: dwillis@dynamicsoft.com

Robert J. Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

email: rsparks@dynamicsoft.com

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

email: bcampbell@dynamicsoft.com

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

email: schulzrinne@cs.columbia.edu

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

email: lennox@cs.columbia.edu

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

email: huitema@microsoft.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

email: bernarda@microsoft.com

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

email: dgurle@microsoft.com

David Oran
Cisco Systems
170 West Tasman Dr.
San Jose, CA 95134

email: oran@cisco.com

Appendix A. Requirements Evaluation

This section was moved forward verbatim from -00.

RFC 2779 [3] outlines requirements for IM and presence protocols. The document describes both shared requirements and IM and presence specific requirements. Examining each of the IM requirements in turn, we also observe that they are met by this proposal:

"Requirement 2.1.1: The protocols MUST allow a PRESENCE SERVICE to be available independent of whether an INSTANT MESSAGE SERVICE is available, and vice-versa." This requirement is met by the separation of presence and IM which we propose here.

"Requirement 2.1.2. The protocols must not assume that an INSTANT INBOX is necessarily reached by the same IDENTIFIER as that of a PRESENTITY. Specifically, the protocols must assume that some INSTANT INBOXes may have no associated PRESENTITIES, and vice versa." This requirement is also easily met by any architecture which completely separates IM and presence as we propose.

"Requirement 2.1.3. The protocols MUST also allow an INSTANT INBOX to be reached via the same IDENTIFIER as the IDENTIFIER of some PRESENTITY." Same as above.

"Requirement 2.1.4. The administration and naming of ENTITIES within a given DOMAIN MUST be able to operate independently of actions in any other DOMAIN." This requirement is met by SIP. SIP uses email-like identifiers which consist of a user name at a domain. Administration of user names is done completely within the domain, and these user names have no defined rules or organization that needs to be known outside of the domain in order for SIP to operate.

"Requirement 2.1.5. The protocol MUST allow for an arbitrary number of DOMAINS within the NAMESPACE." This requirement is met by SIP. SIP uses standard DNS domains, which are not restricted in number.

"Requirement 2.2.1. It MUST be possible for ENTITIES in one DOMAIN to interoperate with ENTITIES in another DOMAIN, without the DOMAINS having previously been aware of each other." This requirement is met by SIP, as it is essential for establishing

sessions as well. DNS SRV records are used to discover servers for a particular service within a domain. They are a generalization of MX records, used for email routing. SIP defines procedures for usage of DNS records to find servers in another domains, which include SRV lookups. This allows domains to communicate without prior setup.

"Requirement 2.2.2: The protocol MUST be capable of meeting its other functional and performance requirements even when there are millions of ENTITIES within a single DOMAIN." Whilst it is hard to judge whether this can be met by examining the architecture of a protocol, SIP has numerous mechanisms for achieving large scales of users within a domain. It allows hierarchies of servers, whereby the namespace can be partitioned among servers. Servers near the top of the hierarchy, used solely for routing, can be stateless, providing excellent scale.

"Requirement 2.2.3: The protocol MUST be capable of meeting its other functional and performance requirements when there are millions of DOMAINS within the single NAMESPACE." The usage of DNS for dividing the namespace into domains provides the same scale as todays email systems, which support millions of DOMAINS.

"Requirement 2.3.5: The PRINCIPAL controlling an INSTANT INBOX MUST be able to control which other PRINCIPALS, if any, can send INSTANT MESSAGES to that INSTANT INBOX." This is provided by access control mechanisms, outside the scope of this extension.

"Requirement 2.3.6: The PRINCIPAL controlling an INSTANT INBOX MUST be able to control which other PRINCIPALS, if any, can read INSTANT MESSAGES from that INSTANT INBOX." This is accomplished through authenticated registration requests. Registrations are used to determine which user gets delivered an instant message. Policy in proxies can allow only certain users to register contact address for a particular inbox (an inbox is defined by the address-of- record in the To field in the registration).

"Requirement 2.4.3: The protocol MUST allow the sending of an INSTANT MESSAGE both directly and via intermediaries, such as PROXIES." This is fundamental to the operation of SIP.

"Requirement 2.4.4: The protocol proxying facilities and transport practices MUST allow ADMINISTRATORS ways to enable and disable protocol activity through existing and commonly-deployed FIREWALLS. The protocol MUST specify how it can be effectively filtered by such FIREWALLS." Although SIP itself runs on port 5060 by default, any other port can be used. It is simple to specify that IM should run on a different port, if so desired.

"Requirement 2.5.1. The protocol MUST provide means to ensure confidence that a received message (NOTIFICATION or INSTANT MESSAGE) has not been corrupted or tampered with." This is supported by SIPs PGP and S/MIME authentication mechanism.

"Requirement 2.5.2. The protocol MUST provide means to ensure confidence that a received message (NOTIFICATION or INSTANT MESSAGE) has not been recorded and played back by an adversary." This is provided by SIP's challenge response authentication mechanisms, through timestamp-based replay prevention, or through stateful storage of previous transaction identifiers (the combination of To, From, Call-ID, CSeq).

"Requirement 2.5.3. The protocol MUST provide means to ensure that a sent message (NOTIFICATION or INSTANT MESSAGE) is only readable by ENTITIES that the sender allows." This is supported through SIPs end to end and hop by hop encryption mechanisms.

"Requirement 2.5.4. The protocol MUST allow any client to use the means to ensure non-corruption, non-playback, and privacy, but the protocol MUST NOT require that all clients use these means at all times." All algorithms for security in SIP are optional.

"Requirement 4.1.1. All ENTITIES sending and receiving INSTANT MESSAGES MUST implement at least a common base format for INSTANT MESSAGES." We specify text/plain here.

"Requirement 4.1.2. The common base format for an INSTANT MESSAGE MUST identify the sender and intended recipient." This is accomplished with the To and From fields in SIP.

"Requirement 4.1.3. The common message format MUST include a return address for the receiver to reply to the sender with another INSTANT MESSAGE." This is done through the Contact headers defined in SIP.

"Requirement 4.1.4. The common message format SHOULD include standard forms of addresses or contact means for media other than INSTANT MESSAGES, such as telephone numbers or email addresses." SIP supports any URL format in the Contact headers. Furthermore, the body of a MESSAGE request can be multipart, and contain things like vCards.

"Requirement 4.1.5. The common message format MUST permit the encoding and identification of the message payload to allow for non-ASCII or encrypted content." MIME content labeling is used in SIP.

"Requirement 4.1.6. The protocol must reflect best current

practices related to internationalization." SIP uses UTF-8 and is completely internationalized.

"Requirement 4.1.7. The protocol must reflect best current practices related to accessibility." Additional requirements are needed on what is required for accessibility.

"Requirement 4.1.9. The working group MUST determine whether the common message format includes fields for numbering or identifying messages. If there are such fields, the working group MUST define the scope within which such identifiers are unique and the acceptable means of generating such identifiers." This is done with the combination of Call-ID and CSeq. The mechanisms for guaranteeing uniqueness are specified in SIP.

"Requirement 4.1.10. The common message format SHOULD be based on IETF-standard MIME (RFC 2045)[14]." SIP uses MIME.

"Requirement 4.2.1. The protocol MUST include mechanisms so that a sender can be informed of the SUCCESSFUL DELIVERY of an INSTANT MESSAGE or reasons for failure. The working group must determine what mechanisms apply when final delivery status is unknown, such as when a message is relayed to non-IMPP systems." SIP specifies notification of successful delivery through 200 OK. When delivery of requests through gateways, success can be indicated only through the SIP component (if the gateway acts as a UAS/UAC) or through the entire system (if it acts like a proxy).

"Requirement 4.3.1. The transport of INSTANT MESSAGES MUST be sufficiently rapid to allow for comfortable conversational exchanges of short messages." The support for end to end messaging (i.e., without intervening proxies) allows IMs to be delivered as rapidly as possible. The UDP reliability mechanisms also support fast recovery from loss.

Full Copyright Statement

Acknowledgement

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